TOWARDS MORE INTELLIGENT EXTRACTION OF INFORMATION FROM DOCUMENTS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Xinya Du

August 2021

TOWARDS MORE INTELLIGENT EXTRACTION OF INFORMATION FROM DOCUMENTS

Xinya Du, Ph.D.
Cornell University 2021

The speed with which new online content becomes available has exacerbated the well known problem of information overload and motivated the innovation of techniques that help people read and consume information. More specifically for NLP, it has motivated researchers to design general models that can extract from large collections of documents informative structured information, such as information about events – what's happening around the world. The extracted structured information (e.g., participants, locations, objects involved in an event) is essential for a variety of downstream tasks such as knowledge base population, question answering, and document analysis.

In recent years, with the progress in research on deep learning, the community has seen improvements on many sentence-level information extraction tasks such as named entity recognition and relation extraction. But less progress has been made on document-level extraction problems (where the elements to be extracted are spread across the document), despite the fact that document-level extraction is closer to what is needed by end users. Existing methods largely ignore the document-level context and split the full extraction problem into separate tasks which cause error propagation. Plus, they rely heavily on manually annotated resources developed for a fixed domain-specific output schema, and, as a result, are not data-efficient or general enough to handle unanticipated schema changes at deployment time.

In this dissertation, we introduce models and frameworks to address these shortcomings of prior work. To better incorporate the document-level context we propose a multi-granularity machine reader, which interprets sentences in the context of preceding and following sentences. To help neural network-based models better capture the output structure and dependencies between events, we propose a generative learning-based framework for the extraction problem, which tackles this complicated task in one pass, avoiding error propagation introduced by traditional pipeline-based systems. Finally, we formulate the (zero-shot) event extraction problem as a question answering task and develop a question answering-based framework, to allow the model to conduct extraction for roles given few/no annotated examples. To further exploit the advantages of the QA-based framework, we propose a learning-based method that automatically generates synthetic question-answer pairs for data augmentation purposes.

BIOGRAPHICAL SKETCH

Xinya Du was born in Jiangsu, China. He developed an interest in mathematics and language studies at an early age. Since high school, he found the field of competitive programming and algorithm design fascinating. This motivated him to choose computer science as his major in the undergraduate institution – Shanghai Jiao Tong University (SJTU) since 2012. During the last year of his undergraduate study, he did a research internship in 2015 with Prof. Claire Cardie in deep learning for fine-grained opinion analysis. This experience greatly motivated him to choose natural language processing and machine learning as a future research direction and to enroll in a Ph.D. program.

After finishing his bachelor's degree at SJTU, he came to Cornell to pursue a doctorate in computer science with a focus on research in the field of natural language processing. During his Ph.D., he focused on research in the fields of question generation, question answering, and information extraction, with the goal to develop better machine reading systems that are capable of turning documents into a structured and concise format and uncovering knowledge beyond direct facts. He also had several fruitful internship experiences at Google AI (during the summer of 2020), Allen Institute for Artificial Intelligence (during the fall of 2018), and Microsoft Research Redmond (during the summer of 2018).

This dissertation is dedicated to my parents.

ACKNOWLEDGEMENTS

First and foremost, I am thankful to my advisor Claire Cardie for her guidance, support, trust, and all the mentoring and help she provided. I also like to express thanks to Prof. John Hopcroft for being on the thesis committee, and especially for his higher-level advice on career and life during our conversations. I like to thank Prof. Immanuel Trummer and Prof. Adrian Sampson for their insightful questions on my research during the oral exams. I was also fortunate to collaborate on research with Prof. Sasha Rush after he joined Cornell Tech. He provides me with a different perspective of NLP research and I learned a lot from him. The work done under his and Claire's guidance is an important part of this dissertation.

I want to thank my mentors for internships during my Ph.D.: Paul Bennett, Ahmed Hassan Awadallah at Microsoft Research; Bhavana Dalvi, Peter Clark at Allen Institute for Artificial Intelligence; Luheng He, Yuan Zhang, Panupong Pasupat and Qi Li at Google.

I would like to thank my peers in Claire's Crew for suggestions and helpful conversations: Ozan Irsoy, Vlad Niculae, Xilun Chen, Arzoo Katiyar, Yao Cheng, Esin Durmus, Kai Sun, Menglin Jia, Rishi Bommasani, Jialu Li. Also, many thanks to students in Cornell NLP groups for helpful discussions/feedback: Tianze Shi, Ana Smith, Liye Fu, Justin Chiu.

Last but not least, I thank my parents, for listening to me patiently when I feel under pressure. During Covid-19, I'm not able to fly back to China to be with them physically, but my heart is always with them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

Over the years, how people absorb information has changed dramatically. For example, in the 1950s, without the existence of the Internet, almost no information was available online. While nowadays, with the explosion of information online, many personal devices (e.g., cell phones, tablets, laptops) are pushing what's happening around the world to people virtually 24 hours a day. This amount of information is over the bar that one can consume, which causes the well-known phenomenon called *information overload*. This phenomenon has motivated research into building better "machine readers" that can automatically process large amounts of textual information. They read through the documents sentence by sentence and extract information necessary for downstream tasks like automatic question answering and knowledge-base population as well as supporting higher-level applications in areas like personal decision-making and government policy making (case/legal documents).

To facilitate downstream tasks like answering user questions or adding new knowledge/facts into a knowledge base to tackle information overload problems in people's lives, researchers have introduced the task called *Information Extraction* (IE). More formally, IE is the automatic identification and classification of instances of user-specified types of entities (e.g., names of places, people, organizations), relations (e.g., person-X is-parent-of person-Y), and events (e.g., an event trigger word and the corresponding entities involved) from text [Grishman, 2019]. And the output is of a structured format (e.g., database) that is machine-readable.

An example of this will be provided later in Figure 1.1.

My goal in this dissertation is to improve machine learning-based methods for extracting structured information from documents and to develop new methods that achieve decent performance given few annotated training examples.

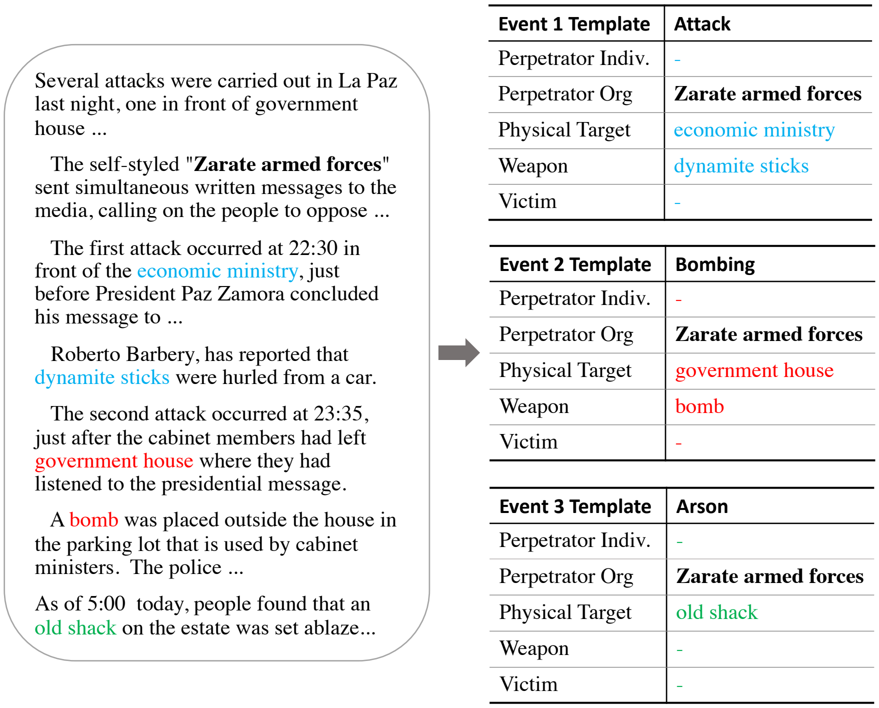
1.2 Information Extraction from Documents

In this dissertation, we focus on extracting information at the document level. The templates to be extracted have a complex structure and the entities/relations involved in each template might be scattered across the document. To be more specific, given a domain of interest, the goal is to extract from it important contents like the happening of certain events. Consider the example from Figure 1.1 in which the input is an article regarding a terrorist incident. In the task, a set of role names are pre-defined (i.e., representing a fixed schema) for the domain, which is terrorism. For example, the roles include `perpetrator-individual` and `weapon`. Also in the task, a set of possible event types (e.g., `Attack`, `Bombing`) are pre-specified.

The target is to build systems that can automatically identify all events in the domain and extract their types and role-fillers in the form of a structured template, one for each distinct event in the input text[1]. There might be zero or multiple (>= 1) templates extracted from each document and zero or multiple role-filler entities to be extracted for each role.

This document-level extraction problem is generally decomposed into two steps,

[1] For the example in Figure 1.1, the system is expected to extract three templates – one for an `Attack`, another for a `Bombing`, the last for an `Arson`.



Several attacks were carried out in La Paz last night, one in front of government house ...

The self-styled "**Zarate armed forces**" sent simultaneous written messages to the media, calling on the people to oppose ...

The first attack occurred at 22:30 in front of the economic ministry, just before President Paz Zamora concluded his message to ...

Roberto Barbery, has reported that dynamite sticks were hurled from a car.

The second attack occurred at 23:35, just after the cabinet members had left government house where they had listened to the presidential message.

A bomb was placed outside the house in the parking lot that is used by cabinet ministers. The police ...

As of 5:00 today, people found that an old shack on the estate was set ablaze...

Event 1 Template	**Attack**
Perpetrator Indiv.	-
Perpetrator Org	**Zarate armed forces**
Physical Target	economic ministry
Weapon	dynamite sticks
Victim	-

Event 2 Template	**Bombing**
Perpetrator Indiv.	-
Perpetrator Org	**Zarate armed forces**
Physical Target	government house
Weapon	bomb
Victim	-

Event 3 Template	**Arson**
Perpetrator Indiv.	-
Perpetrator Org	**Zarate armed forces**
Physical Target	old shack
Weapon	-
Victim	-

Figure 1.1: Extracting Structured Information from Documents (Template Filling).

1. Event/trigger detection: In the event detection step, the system should read through the input sequence to determine/detect the existence of events (e.g., a trigger word or expression).

2. Role-filler entity extraction[2]: In the role-filler entity extraction phase, the system should extract entity spans corresponding to the roles associated with the template type.

Generally, to extract the correct information from a document, a system needs to model the entire document and its complex output structure well [Ji and Grishman, 2008; Liao and Grishman, 2010; Yang and Mitchell, 2016]. One difficulty is that, from a single input sentence alone, it is often hard/ambiguous to determine the type of the event. For example, "left" might mean someone left a place or

[2] Under the ACE [Linguistic Data Consortium, 2005] setting, this is called argument extraction.

ended a position, depending on the document-level context. Another difficulty is that there are often more than multiple interrelated events described in the document, and jointly extracting them and modeling the structure between them while avoiding error propagation is important.

However, there are two main challenges for document-level information extraction that are not tackled by prior methods:

- **Challenge I: Document-level Context and Complex Output Structure.** Ignoring document-level **context** is problematic when the information needed to recognize an event argument is spread across multiple sentences. For the example below, determining whether to extract "government house" in sentence 1 as a role-filler entity of type `physical-target` depends on the additional context in sentence 2.

  > [S1] The second attack occurred at 23:35, just after the cabinet members had left [*government house*] $_{\texttt{physical-target}}$ where they had listened to the presidential message.
  >
  > [S2] A [bomb] $_{\texttt{weapon}}$ was placed outside *the house* in the parking lot that is used by cabinet ministers.

  One approach to incorporate the contextual information is through coreference resolution (CR) [Yang and Mitchell, 2016] – the CR system would first connect "government house" in S1 with "the house" in S2. But this approach typically suffers from error propagation due to multiple stages of processing.

  What's more, the **structure** of the required extractions from an entire document is quite complex in comparison to the sentence-level extraction: the number of frames is not pre-determined (e.g., for the document in Figure 1.1, a fourth event might need to be extracted if the article continues) and the information associated with each event frame can be dispersed throughout the document. To conduct the extraction correctly, the system needs to capture within-event structure (i.e., the dependency between entities of different roles) – for example, “bomb” is more often used in a bombing event; entity “house” of role `physical-target` and “bomb” of role `weapon` might appear together often. In addition, capturing cross-event dependency is important – the same entity might be responsible for multiple events across the document even though it is mentioned only once (e.g., **Zarate armed forces** in Figure 1.1 is of role `perpetrator-organization` across all templates). However, current end-to-end learning-based systems, in spite of their high accuracy, ignore this structured information and often make inconsistent predictions at test time [Wadden et al., 2019; Du et al., 2021a].

- **Challenge II: High Cost and Limited Availability of Annotations.** Annotations for document-level extraction problems are harder and more costly to obtain as compared to sentence-level extraction tasks. It is time-consuming and usually requires people with domain expertise [Jain et al., 2020; Li et al., 2021]. Another relevant challenge faced by current IE systems' formulation is that: when systems are trained on a limited amount of resources with a fixed schema of roles (e.g., event frames with only `organization`, `recipient`, `outcome` and, `time`), they are trained in a discriminative way – predicting roles of the detected argument spans. Under this formulation, they are not able to handle unseen but relevant roles (e.g., `participant`, `place`) at deployment time.

Motivated by these major challenges, my research work in the area of information extraction and question generation tackles them from different perspectives. They are described later in this dissertation.

1.3 Contributions

The main contribution of this dissertation is the development of context-aware end-to-end methodologies for extracting structured information from textual documents. More specifically, we make the following contributions:

Multi-Granularity Contextualized Encoding of Document Context. Few works in the literature of information extraction have gone beyond individual sentences to make extraction decisions. This is problematic when the information needed to make an extraction decision is spread across multiple sentences. To mitigate the problem, we propose a novel multi-granularity reader for document-level role-filler extraction (a sub-task of template filling). When reading through the documents to make extraction decisions, our approach dynamically aggregates information captured by neural representations learned at different levels of granularity (e.g., the sentence- and paragraph-level). In this work, we also investigate how the length of maximum context captured affects the models' performance. More information on the techniques and experiments is explained in Chapter 3.

Generative Transformers for Document-level Template Filling. The complexity of the document-level extraction tasks comes not only from the need to keep a potentially lengthy context in mind, but also due to the complex structure of the desired output (i.e., *multiple* templates that are inter-related are to be extracted; within each template, entities of different roles are dependent). In consideration of this,

1. We introduce a generative transformer-based encoder-decoder framework (GRIT) for role-filler **entity** extraction (REE) that is designed to model

the context at the document level: it can make extraction decisions across sentence boundaries; is implicitly aware of noun phrase coreference, and can respect within-event cross-role dependencies in the template;

2. Further, we extend the GRIT model to also handle the full task of template filling (GTT) – GTT goes beyond extracting information for a single event and handles the case where multiple events/templates are described in the document. We demonstrate that the sequence-to-sequence learning setup is good at capturing cross-event structure when doing predictions.

The GRIT and GTT approaches and experiments will be described in detail in Chapter 4.

Formulating Event Extraction as Question Answering. Prior methods in the IE literature rely heavily on entity information/annotations and are unable to extract fillers for event argument roles that are not seen during training. Plus, pipeline-based methods have generally been used for event extraction during decoding – they generally perform *trigger detection* → *entity recognition* → *argument role assignment*. Instead, we introduce a new paradigm for event extraction by formulating it as a question answering (QA) task that extracts the event arguments in an end-to-end manner (Chapter 5). We design a set of increasingly natural question generation templates to better access "knowledge" implicitly encoded the pre-trained language models during this process. We also show that our framework is capable of extracting event arguments for roles not seen at training time (i.e., in a zero-shot learning setting). In Chapter 5, more information about this QA-driven framework can be found.

Generation of Paragraph-Level Question-Answer Pairs. To help mitigate the problem of limited annotation availability and help with data augmentation, we study the task of generating from Wikipedia articles question-answer pairs that cover content beyond a single sentence. We propose a neural network approach that incorporates coreference knowledge via a novel gating mechanism. Compared to models that only take into account sentence-level information, we find that the linguistic knowledge introduced by the coreference representation aids question generation significantly, producing models that outperform the current state-of-the-art. As shown by follow-up work in the literature of information extraction, the synthetic QA-pairs generated by machine learning models can help boost the performance of IE systems.

1.4 Roadmap

The remainder of this dissertation is organized as follows. Firstly in Chapter 2, we'll provide background on task definitions and an overview of the relevant machine learning-based approaches, especially approaches based on neural networks. We introduce our work on multi-granularity contextualized encoding for modeling document-level context for extracting event role-filler *mentions* in Chapter 3. In Chapter 4, we first present the work on a generative end-to-end transformer-based model for extracting event *entities*; then we continue into the details on how to extend it to do hierarchical decoding for the case where there are multiple events/templates in a single document (i.e., template filling). In Chapter 5, we demonstrate how to build QA-driven models that operate by accessing knowledge embedded in pre-trained neural models and allow the introduction of new roles at test time. In Chapter 6, we present our question generation framework for automatically generating/harvesting synthetic QA pairs that has been shown to be beneficial for the event extraction task when used to provide additional training data.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we first present an overview of the information extraction problem, and existing research on sentence-level extraction and document-level extraction tasks. We then present existing branches of neural network-based methods that are related to our work presented in this dissertation. In later chapters, we will discuss our proposed models and analysis in detail.

2.1 Information Extraction

The literature on information extraction (IE) dates back to close 1994 when the Advanced Research Projects Agency (ARPA) begin to support research to develop the "new technology" called IE [Okurowski, 1993]. The goal was to tackle the increasing demands on processing and analyzing large volumes of online data. MUC conferences (Message Understanding Conference) first defined several extraction tasks called "template filling". The target is to fill a predefined "template" (representing a stereotypical event or situation) with information directly extracted from the document, as well as concepts like amounts, or ontology entities that have to be inferred through additional processing. Among the evaluations, MUC-1 and MUC-2 used Navy exercise message traffic as the corpus; MUC-3 and MUC-4's input articles are news about terrorism in Latin America [Chinchor et al., 1993]. Methods developed for these tasks at that time involved a long pipeline and relied on manually designed specific patterns for each domain [Grishman, 2012].

After a series of MUC evaluations (i.e., MUC-1 to MUC-7), it becomes clear that annotating training data and supervised training is effective for improving

IE systems' performance. In addition, a working group was formed which recommended extracting a set of elementary events and their arguments rather than a monolithic template. Under such considerations, Automatic Content Extraction (ACE) evaluation [1] in 2005 provided a larger collection of news articles across three languages (i.e., English, Chinese, and Arabic). Each document is represented by a set of entities (7 types), relations (6 types), and events (8 types). One major difference to MUC is that: the arguments to a relation or event must occur in *one* single sentence under the ACE formulation. Extracting the **named entities** involves identifying and classifying all the names in the corpus. After grouping entity mentions which are coreferential and assigning each group a semantic type in the schema, people have the **entities**. In addition, after extraction for entities, people can determine the **relations** between pairs of entities (e.g., `part-whole`, `personal-social`). The extraction of an **event** includes the identification of the trigger word — the main word that describes the event, and the extraction of event arguments with roles (each type of event has a predefined set of argument roles based on the annotation guideline).

Next, we'll mainly focus on the perspective of sentence-level and document-level (event) extraction, and introduce more details including various tasks and datasets in both subsections. We provide a brief summary and comparison between different representative IE datasets in Table 2.1.

	Document-level	Entity Mentions	Relations	Events	Multiple Events
MUC-3,4	✓			✓	✓
ACE05		✓	✓	✓	✓
GENIA [Kim et al., 2003]		✓	✓	✓	✓
TACRED [Zhang et al., 2017]		✓	✓		
SciERC [Luan et al., 2018]		✓	✓		
SciREX [Jain et al., 2020]	✓	✓	✓		
DocRED [Yao et al., 2019]	✓	✓	✓		
RAMS [Ebner et al., 2020]	✓			✓	
WikiEvents [Li et al., 2021]	✓			✓	✓

Table 2.1: Comparison of Representative IE Datasets.

2.1.1 Sentence-level Extraction Tasks

As we mentioned previously, in the ACE05 [Doddington et al., 2004; Walker et al., 2006][2] dataset, arguments to a relation or event must occur in *one* single sentence. For the example below, the input sentence contains one event triggered by the word "sale". All the annotated arguments of the event (e.g., "operations", "French company") are within the same sentence.

> ... As part of the 11-billion-dollar **sale** of USA Interactive's film and television [operations]$_{\texttt{Artifact}}$ to the [French company]$_{\texttt{Buyer}}$ and its [parent company]$_{\texttt{Buyer}}$ in December 2001, [USA Interactive]$_{\texttt{Seller}}$ received 2.5 billion dollars in preferred shares in Vivendi Universal Entertainment.

Notice that, in the ACE dataset, the input sentence is in a document and additional document-level context could still be leveraged. In addition, the input sentence might contain multiple events represented by different trigger words.

As for entity mention and relation extraction, apart from the ACE dataset, Zhang et al. [2017] introduce the TACRED dataset, which is obtained via crowd-

[1] http://www.itl.nist.gov/iad/mig/tests/ace/2005/doc/ace05-evalplan.v2a.pdf

[2] catalog.ldc.upenn.edu/LDC2006T06

sourcing and targeted towards TAC KBP relations (reusing the entity and relation types of the TAC KBP tasks). Specifically for in the scientific domain, SciERC [Luan et al., 2018] dataset is collected from 500 AI-related papers' abstracts and defines scientific terms and relations for knowledge graph construction. It is with annotations for scientific entities and their relations. The GENIA[3] dataset [Kim et al., 2003] consists of sentences from the molecular biology domain and includes term annotations (e.g., proteins, genes, and cells) and relation annotations (e.g., protein-protein interactions).

2.1.2 Document-level Extraction Tasks

Document-level extraction dates back to the template filling tasks from the MUC conferences [Grishman and Sundheim, 1996] for specific scenarios. The evaluations required extracting string fillers (e.g., participating entities) that can be directly found in the text and categorical fillers (e.g., attribute values, amounts, effect on the event target) that need to be inferred. In this dissertation, our works using the MUC datasets only tackle the extraction of string fillers. Readers can refer to Figure 1.1 for a general idea of what the template filling task looks like. To find the full template definition, readers can refer to Grishman [2019]. In contrast to sentence-level event extraction, the document-level template filling task introduces several complications:

- There is no explicit event trigger annotation in this formulation. Directly grouping the participants together into multiple templates involves more stages (e.g., the system needs to first identify how many events there are

[3] http://www.geniaproject.org/genia-corpus

in the document).

- Role-filler entities for the same template/event spread across multiple sentences, determining their relations (i.e., whether two entities are involved in the same event) becomes harder as compared to the sentence-level extraction.
- To completely get the extractions correct, the system also needs to model the coreference structure to avoid spurious role-filler entity mention extractions.

Recently, there have been newer datasets such as RAMS [Ebner et al., 2020] and WikiEvents [Li et al., 2021] for document-level event extraction. More specifically, Ebner et al. [2020] found that 38.1% of the annotated events in AIDA-1 have an argument outside the sentence containing the trigger and they release RAMS, which contains annotations for cross-sentence implicit arguments covering a wide range of event types. However, their dataset only annotates one event for each document. WikiEvents has complete event and coreference annotations for each document, based on the ontology from the KAIROS project[4]. For datasets of languages other than English, Zheng et al. [2019] release a document-level event extraction dataset Doc2EDAG, which contains Chinese financial announcements.

Apart from event extraction, SciREX [Jain et al., 2020] annotates document-level N-ary relations[5] for scientific articles with both automatic and human annotation methods. Besides, there are some document-level N-ary relation extraction datasets constructed via only distant supervision from a small number of manually curated facts [Quirk and Poon, 2017; Peng et al., 2017], but inevitable labeling errors often occur in them. Specifically for pairwise relation extraction, Yao et al. [2019] annotated DocRED (via crowdsourcing) for document-level relation extraction between entities.

[4] https://www.ldc.upenn.edu/collaborations/currentprojects

[5] Extracting relations between more than two arguments.

2.2 End-to-End Learning-based Methods

Researchers also tried to investigate how to apply end-to-end learning models to IE tasks and bypass the process of heavy feature engineering (i.e., building a set of linguistic features for each task). Neural network-based models have been demonstrated to be effective on these IE tasks, achieving state-of-the-art performance on most of the benchmarks. In this section, we focus on reviewing relevant branches of models that are most relevant to this dissertation. More specifically, we first provide a basic background for contextualized word representations provided by pre-trained language models that are mostly treated as a base component of the end-to-end IE models. Then, we review several representative branches of modeling choices using neural networks for entity mentions and argument span extractions.

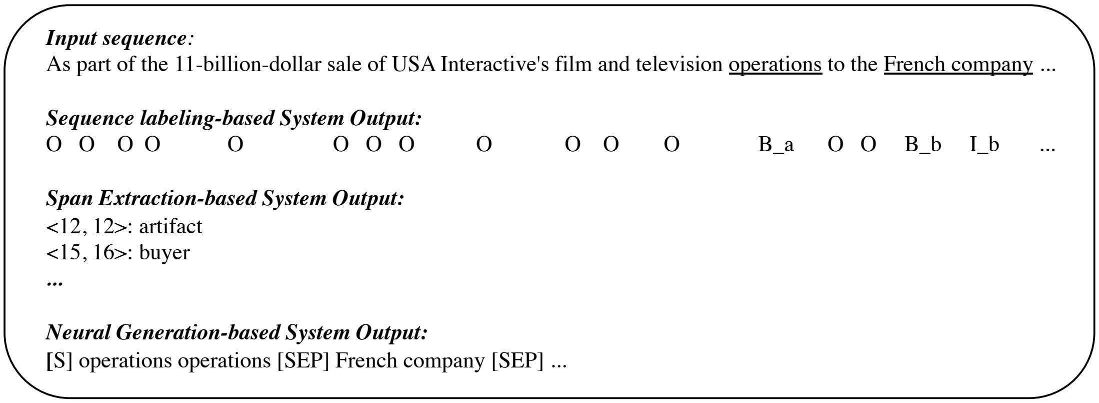
2.2.1 (Contextualized) Word Representations

Around 2013, researchers investigated methods for learning good word embeddings that could be for improving downstream tasks (e.g., Skip-gram [Mikolov et al., 2013] and GloVe [Pennington et al., 2014]). These embeddings capture the semantic similarity of words by embedding all words in a dense low-level representation. They are called *distributed representations* as compared to the "one-hot" representation of words in the previous feature-based approaches. The pre-trained embeddings have been proved to be effective for many NLP tasks like named entity recognition and document classification. Although being powerful, they are *context-free* and static and fail to capture higher-level concepts in the context, such as semantic roles, syntactic structures, and anaphora [Qiu et al., 2020]. Later there has been a trend of research on learning *contextualized word representations*.

These efforts include models based on transformers [Vaswani et al., 2017a] like GPT-2 [Radford et al., 2018] and BERT [Devlin et al., 2019]; as well as models based on LSTMs [Hochreiter and Schmidhuber, 1997] like ELMo [Peters et al., 2018]. These models are pre-trained on a large amount of unlabeled free text via different training objects. For downstream tasks, people usually fine-tune the pre-trained models on the target dataset, and words are represented in the context of other words (dynamic embedding). Most later research for many NLP tasks has reported the advantages of contextualized word representations over context-free word embeddings. In addition, it has been shown that pre-trained language models are better at capturing long-range dependencies [Wadden et al., 2019], thus benefiting from cross-sentence context.

Our models to be introduced in this dissertation for IE problems are based on contextualized word representations provided by pre-trained models.

2.2.2 Modeling Choices for Entity/Argument Extraction



***Input sequence**:*
As part of the 11-billion-dollar sale of USA Interactive's film and television operations to the French company ...

Sequence labeling-based System Output:
```
O  O  O O      O       O  O  O     O      O  O    O       B_a   O  O  B_b  I_b   ...
```

Span Extraction-based System Output:
<12, 12>: artifact
<15, 16>: buyer
...

Neural Generation-based System Output:
[S] operations operations [SEP] French company [SEP] ...

Figure 2.1: Output Format for Different Neural Network-based Methods.

Next, we review several branches of modeling choices based on neural networks for entity mention/argument span extraction – the traditional BIO sequence labeling-based model, the span-based extraction model, and the more recent neural generation-based models. For better intuitive comparison, we provide their corresponding output formats in Figure 2.1.

Sequence Labeling-based Extraction Approaches Entity mention recognition and argument span extraction have been commonly treated as a sequence labeling problem. For the input sequence in the example, models predict the corresponding tag for each token in it – B_x, I_x or O. The prefix "B" marks the beginning of a span, and "I" means inside of a span. A token not belonging to any span is tagged with "O". "x" stands for the specific type for the span. The sequence labeling-based models have benefited substantially from the use of neural representations [Collobert et al., 2011; Lample et al., 2016]. Later, researchers found that adding the conditional random field component (CRF) on top of the neural network units like LSTM can help with structured learning for sequences – Huang et al. [2015] propose the LSTM-CRF model. On top of that, Ma and Hovy [2016]) found performance could be further improved by incorporating character-level representations of words via CNN (LSTM-CNN-CRF).

Span-based Extraction Approaches Different from the sequence labeling paradigm, models using a span extraction paradigm directly predict the beginning and ending offsets of the gold entity mention from words in the input sequence, as well as the type/role for it. Inspired by end-to-end span-based models for coreference resolution [Lee et al., 2017], DYGIE [Luan et al., 2019] introduces a general framework that share span representations using dynamically built span graphs and shows that the span-based method naturally excels at extracting entities with overlapping spans as compared to sequence-labeling based approach [Katiyar and Cardie, 2018]. DYGIE++[Wadden et al., 2019] extended the DYGIE framework to also extract events. From the technical perspective, it leverages BERT embeddings to build more robust multi-sentence representations.

Inspired by the progress on certain NLP problems when formalizing them as question answering/machine reading comprehension tasks [McCann et al., 2018], Li et al. [2019b,a] convert the named entity recognition & relation extraction task into a multi-turn question answering problem. In their formulation, the model for NER is also span-based – predicting the beginning and ending offsets for the mention entity in the input sequence.

To tackle the error propagation problem caused by pipeline-based systems and making models capable of handling unanticipated roles at deployment time in the low annotation setting, we propose to formulate event extraction as a QA task and introduce a new framework – models answer questions generated with annotation guideline information to extract the event trigger and corresponding argument spans (chapter 5). As compared to the pure span enumeration methods like DYGIE, our framework includes a question generation stage, the generated questions encode informative prior knowledge (e.g., natural language descriptions/definitions for argument roles). In addition, as shown by Liu et al. [2020], the QA-based event extraction framework can also benefit from additional QA pairs or synthetically generated QA pairs. Motivated by the importance of generating synthetic QA pairs, we present a neural network-based framework for the automatic generation of QA pairs from Wikipedia in chapter 6.

Neural Generation-based Approaches As shown in Figure 2.1, the generation-based system output needs to be specifically designed: for the example input, the output sequence consists of the start token ([S]), followed by the entity mentions of the first role `Buyer` ("operations"), followed by the separator token ([SEP]), followed by the entity mentions of the second role ("French company"), etc. The target for the neural generation model is to "generate" the raw output sequence, and based on the sequence we can obtain the extracted entity mentions and event templates at inference time.

T5 [Raffel et al., 2019] is a framework that casts problems such as machine translation and summarization as text-to-text tasks in natural language, leveraging the transfer learning power of a transformer-based language model. But T5 is not tested on more complex structured prediction problems in IE. Paolini et al. [2021] propose a generation-based framework based on T5 and call it *Translation between Augmented Natural Languages (TANL)* and demonstrate its effectiveness on structured prediction tasks like joint entity mention and relation extraction and semantic role labeling.

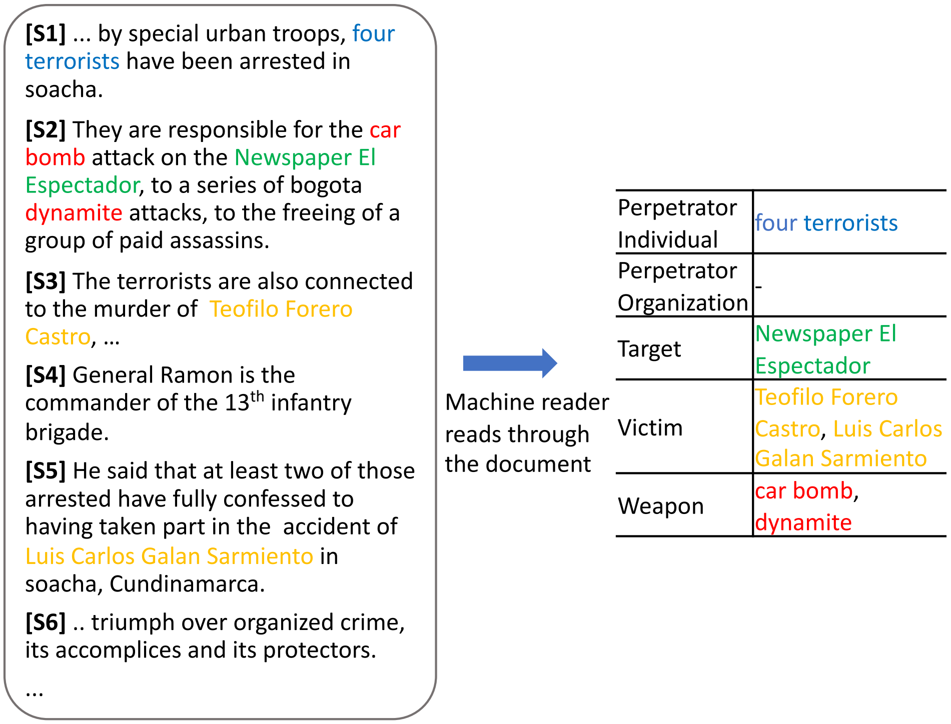
Our models GRIT [Du et al., 2021a] and GTT [Du et al., 2021b] for template filling fall under this branch. Details of them will be introduced in chapter 4.

CHAPTER 3
MULTI-GRANULARITY ENCODING FOR DOCUMENT CONTEXT

In this chapter, we investigate how well end-to-end neural sequence models (with pre-trained language model representations) perform on document-level role filler extraction problem. We propose a novel multi-granularity reader to better capture representations learned at different levels of granularity of the document context (i.e., the sentence- and paragraph-level), instead of making extraction decisions based on only sentence-level context. The work described in this chapter is mainly described in Du and Cardie [2020].

As described in chapter 1, the goal of document-level event extraction[1] is to identify in an article events of a pre-specified type along with their event-specific role fillers, i.e., arguments. The complete document-level extraction problem generally requires *role filler extraction*, *noun phrase coreference resolution* and *event tracking/detection* (i.e., determine which extracted role fillers belong to which event). In this work, we focus only on role filler extraction. Figure 3.1 provides a representative example of this task. Given an article consisting of multiple paragraphs/sentences, and a fixed set of event types (e.g., terrorist events) and associated roles (e.g., Perpetrator Individual, Victim, Weapon), we aim to identify those spans of text that denote the role fillers for all events. This generally requires both sentence-level understanding and accurate interpretation of the context beyond the sentence. Examples include identifying "Teofilo Forero Castro" (mentioned in S3) as a victim of the car bomb attack event (mentioned in S2), determining there's no role filler in S4 (both of which rely mainly on sentence-level

[1] The task is also referred to as *template filling* MUC-4 [1992].



[S1] ... by special urban troops, four terrorists have been arrested in soacha.

[S2] They are responsible for the car bomb attack on the Newspaper El Espectador, to a series of bogota dynamite attacks, to the freeing of a group of paid assassins.

[S3] The terrorists are also connected to the murder of Teofilo Forero Castro, ...

[S4] General Ramon is the commander of the 13th infantry brigade.

[S5] He said that at least two of those arrested have fully confessed to having taken part in the accident of Luis Carlos Galan Sarmiento in soacha, Cundinamarca.

[S6] .. triumph over organized crime, its accomplices and its protectors.

...

Perpetrator Individual	four terrorists
Perpetrator Organization	-
Target	Newspaper El Espectador
Victim	Teofilo Forero Castro, Luis Carlos Galan Sarmiento
Weapon	car bomb, dynamite

Figure 3.1: The document-level event role filler extraction task.

understanding, and identifying “four terrorists” in S1 as a perpetrator individual (which requires coreference knowledge across sentence boundaries).

Recent work in document-level event role filler extraction has employed a pipeline architecture with separate classifiers for each type of role and for relevant context detection [Patwardhan and Riloff, 2009; Huang and Riloff, 2011]. However these methods: (1) suffer from error propagation across different pipeline stages; and (2) require heavy feature engineering (e.g., lexico-syntactic pattern features for candidate role filler extraction; lexical bridge and discourse bridge features for detecting event-relevant sentences at the document level). Moreover, the features are manually designed for a particular domain, which requires linguistic intuition and domain expertise [Nguyen and Grishman, 2015].

Neural end-to-end models have been shown to excel at sentence-level information extraction tasks, such as named entity recognition [Lample et al., 2016; Chiu and Nichols, 2016] and ACE-type within-sentence event extraction [Chen et al., 2015; Nguyen et al., 2016; Wadden et al., 2019]. However, to the best of our knowledge, no prior work has investigated the formulation of document-level event role filler extraction as an end-to-end neural sequence learning task. In contrast to extracting events and their role fillers from standalone sentences, document-level event extraction poses special challenges for neural sequence learning models. First, capturing long-term dependencies in long sequences remains a fundamental challenge for recurrent neural networks [Trinh et al., 2018]. To model long sequences, most RNN-based approaches use backpropagation through time. But it's still difficult for the models to scale to very long sequences. We provide empirical evidence for this for event extraction in the section for experiments. Second, although pretrained bi-directional transformer models such as BERT [Devlin et al., 2019] better capture long-distance dependencies as compared to an RNN architecture, they still have a constraint on the maximum length of the sequence, which is below the length of many articles about events.

In the sections below, we study how to train and apply end-to-end neural models for event role filler extraction. We first formalize the problem as a sequence tagging task over the tokens in a set of contiguous sentences in the document. To address the aforementioned challenges for neural models applied to long sequences, (1) we investigate the effect of context length (i.e., maximum input segment length) on model performance, and find the most appropriate length; and (2) propose a multi-granularity reader that dynamically aggregates the information learned from the local context (e.g., sentence-level) and the broader context (e.g., paragraph-level). A quantitative evaluation and qualitative analysis of our approach on the MUC-4 dataset [MUC-4, 1992] both show that the multi-granularity reader achieves substantial improvements over the baseline models and prior work.

3.1 Related Work

Event extraction has been mainly studied under two paradigms: detecting the event trigger and extracting the arguments from an individual sentence (e.g., the ACE task [Doddington et al., 2004][2], v.s. at the document level (e.g., the MUC-4 template filling task [Sundheim, 1992]).

Sentence-level Event Extraction The ACE event extraction task requires extraction of the event trigger and its arguments from a sentence. For example, in the sentence " ... Iraqi soldiers were *killed* by U.S. artillery ...", the goal is to identify the "die" event triggered by *killed* and the corresponding arguments (PLACE, VICTIM, INSTRUMENT, etc.) Many approaches have been proposed to improve performance on this specific task. Li et al. [2013, 2015] explore various hand-designed features; Nguyen and Grishman [2015]; Nguyen et al. [2016]; Chen et al. [2015]; Liu et al. [2017, 2018] employ deep learning based models such as recurrent neural networks and convolutional neural network. Wadden et al. [2019] utilize pre-trained contextualized representations. The approaches generally focus on sentence-level context for extracting event triggers and arguments and rarely generalize to the document-event extraction setting (Figure 3.1).

Only a few models have gone beyond individual sentences to make decisions. [Ji and Grishman, 2008] enforce event role consistency across documents. [Liao and Grishman, 2010] explore event type co-occurrence patterns to propagate event classification decisions. Similarly, [Yang and Mitchell, 2016] propose *jointly* ex-

[2] https://catalog.ldc.upenn.edu/LDC2006T06

tracting events and entities within a document context. Also related to our work are [Duan et al., 2017b] and [Zhao et al., 2018], which utilize document embeddings to aid event detection with recurrent neural networks. Although these approaches make decisions with cross-sentence information, their extractions are still at the sentence level.

Document-level Event Extraction has been studied mainly under the classic MUC paradigm [MUC-4, 1992]. The full task involves the construction of answer key templates, one template per event (some documents in the dataset describe more than one events). Typically three steps are involved — role filler extraction, role filler mention coreference resolution and event tracking). In this work we only focus on role filler extraction.

From the modeling perspective, recent work explores both the local and additional context to make the role filler extraction decisions. GLACIER [Patwardhan and Riloff, 2009] jointly considers cross-sentence and noun phrase evidence in a probabilistic framework to extract role fillers. TIER [Huang and Riloff, 2011] proposes to first determine the document genre with a classifier and then identify event-relevant sentences and role fillers in the document. Huang and Riloff [2012] propose a bottom-up approach that first aggressively identifies *candidate* role fillers (with lexico-syntactic pattern features), and then removes the candidates that are in spurious sentences (i.e., not event-related) via a cohesion classifier (with discourse features). Similar to Huang and Riloff [2012], we also incorporate both intra-sentence and cross-sentence features (paragraph-level features), but instead of using manually designed linguistic information, our models learn in an automatic way how to dynamically incorporate learned representations of the article. Also, in contrast to prior work that is pipeline-based, our approach tackles the task as an end-to-end sequence tagging problem.

There has also been work on unsupervised event schema induction [Chambers and Jurafsky, 2011; Chambers, 2013] and open-domain event extraction [Liu et al., 2019] from documents: the main idea is to group entities corresponding to the same role into an event template. Our models, on the other hand, are trained in supervised way and the event schemas are pre-defined.

Apart from event extraction, there has been increasing interest on cross-sentence relation extraction Mintz et al. [2009]; Peng et al. [2017]; Jia et al. [2019]. This work assumes that mentions or entities are provided, and thus is more of a pairwise mention/entity-level classification problem. Our work instead focuses on role filler/span extraction using sequence tagging approaches; role filler type is determined during this process.

Capturing Long-term Dependencies for Neural Sequence Models When training neural sequence models such as RNNs, capturing long-term dependencies in sequences remains a fundamental challenge [Trinh et al., 2018]. Most approaches use backpropagation through time (BPTT) but it is difficult to scale to very long sequences. Many variations of models have been proposed to mitigate the effect of long sequence length, such as Long Short Term Memory (LSTM) Networks [Hochreiter and Schmidhuber, 1997; Gers et al., 1999; Graves, 2013] and Gated Recurrent Unit Networks [Cho et al., 2014]. Transformer based models [Vaswani et al., 2017b; Devlin et al., 2019] have also shown improvements in modeling long text. In our work for document-level event role filler extraction, we also implement LSTM layers in the models as well as utilize the pre-trained representations provided by the bi-directional transformer model. From an application perspective, we investigate the suitable length of context to incorporate for the neural sequence tagging model in the document-level extraction setting. We also study how to mitigate problems associated with long sequences by dynamically incorporating both sentence-level and longer context (i.e., paragraph-level) representations in the model (Figure 3.3).

3.2 Methodology

In the following we describe (1) how we transform the document into paired token-tag sequences and formalize the task as a sequence tagging problem (Section 3.2.1); (2) architectures of our base k-sentence reader (Section 3.2.2) and multi-granularity reader (Section 3.2.3).

3.2.1 Constructing Paired Token-tag Sequences from Documents and Gold Role Fillers

We formalize document-level event role filler extraction as an end-to-end sequence tagging problem. Figure 3.2 illustrates the general idea. Given a document and the text spans associated with the gold-standard (i.e., correct) fillers for each role, we adopt the BIO (Beginning, Inside, Outside) tagging scheme to transform the document into paired token/BIO-tag sequences.

We construct example sequences of variant context lengths for training and testing our end-to-end k-sentence readers (i.e., the single-sentence, double-sentence, paragraph and chunk readers). By "chunk", we mean the chunk of contiguous

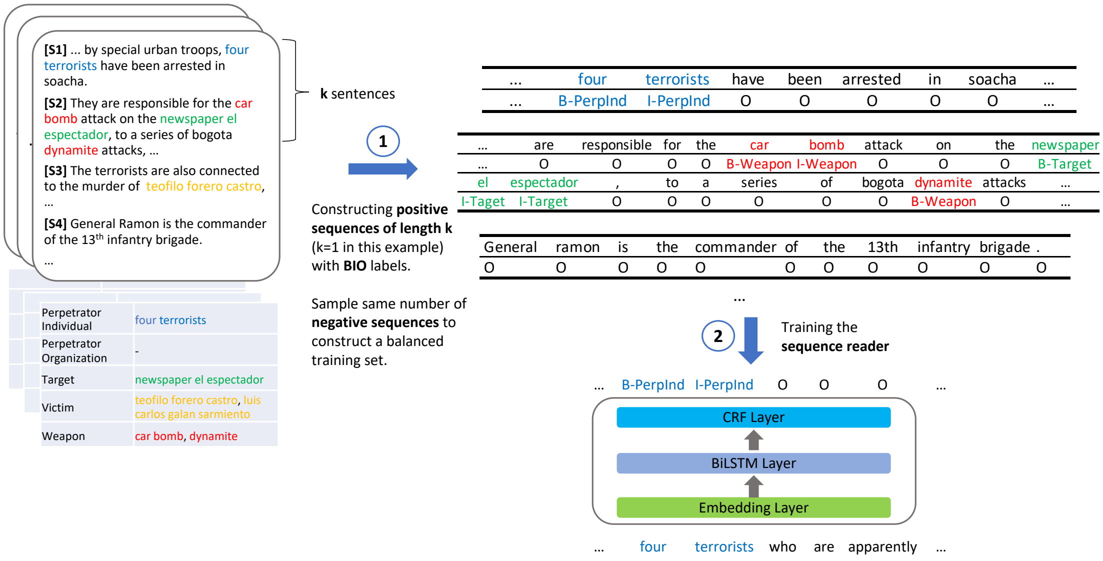

[S1] ... by special urban troops, four terrorists have been arrested in soacha.
[S2] They are responsible for the car bomb attack on the newspaper el espectador, to a series of bogota dynamite attacks, ...
[S3] The terrorists are also connected to the murder of teofilo forero castro, ...
[S4] General Ramon is the commander of the 13th infantry brigade.
...
k sentences
Perpetrator Individual
four terrorists
Perpetrator Organization
-
Target
newspaper el espectador
Victim
teofilo forero castro, luis carlos galan sarmiento
Weapon
car bomb, dynamite
1
Constructing positive sequences of length k (k=1 in this example) with BIO labels.
Sample same number of negative sequences to construct a balanced training set.
... four terrorists have been arrested in soacha ...
... B-PerpInd I-PerpInd O O O O O ...
... are responsible for the car bomb attack on the newspaper
... O O O O B-Weapon I-Weapon O O O B-Target
el espectador , to a series of bogota dynamite attacks ...
I-Taget I-Target O O O O O O B-Weapon O ...
General ramon is the commander of the 13th infantry brigade .
O O O O O O O O O O O
...
2
Training the sequence reader
... B-PerpInd I-PerpInd O O O ...
CRF Layer
BiLSTM Layer
Embedding Layer
... four terrorists who are apparently ...

Figure 3.2: An overview of our framework for training the sequence reader for event role filler extraction.

sentences which is right within the sequence length constraint for BERT – 512 in this case. Specifically, we use a sentence splitter[3] to divide the document into sentences s_1, s_2, ..., s_n. To construct the training set, starting from *each* sentence i, we concatenate the k contiguous sentences (s_i to s_{i+k-1}) to form overlapping candidate sequences of length k: sequence 1 consists of $\{s_1, ..., s_k\}$, sequence 2 consists of $\{s_2, ..., s_{k+1}\}$, etc. To make the training set balanced, we sample the same number of positive and negative sequences from the candidate sequences, where ”positive” sequence contains at least one event role filler, and “negative” sequences contain no event role fillers. To construct the dev/test set for inference, where the reader is applied, we simply group the contiguous k sentences together in order, producing $\frac{n}{k}$ sequences (i.e., sequence 1 consists of $\{s_1, ..., s_k\}$, sequence 2 consists of $\{s_{k+1}, ..., s_{2k}\}$, etc.) For the paragraph reader, we set k to average paragraph length for the training set, and to the real paragraph length for test set.

We denote the token in the sequence with x, the input for the k-sentence reader

[3] https://spacy.io/

is $\mathbf{X} = \{x_1^{(1)}, x_2^{(1)}, ..., x_{l_1}^{(1)}, ..., x_1^{(k)}, x_2^{(k)}, ..., x_{l_k}^{(k)}\}$; where $x_i^{(k)}$ is the i-th token of the k-th sentence, and l_k is the length of the k-th sentence.

3.2.2 k-sentence Reader

Since our general k-sentence reader does not recognize sentence boundaries, we simplify the notation for the input sequence as $\{x_1, x_2, ..., x_m\}$ here.

Embedding Layer In the embedding layer, we represent each token x_i in the input sequence as the concatenation of its word embedding and contextual token representation:

- *Word Embedding*: We use the 100-dimensional GloVe pre-trained word embeddings [Pennington et al., 2014] trained from 6B Web crawl data. We keep the pre-trained word embeddings fixed. Given a token x_i, we have its word embedding: $\mathbf{xe}_i = \mathbf{E}(x_i)$.
- *Pre-trained LM representation*: Contextualized embeddings produced by pre-trained language models [Peters et al., 2018; Devlin et al., 2019] have been proved to be capable of modeling context beyond the sentence boundary and improve performance on a variety of tasks. Here we employ the contextualized representations produced by `BERT-base` for our k-sentence labeling model, as well as the multi-granularity reader to be introduced next. Specifically, we use the average of all the 12 layers' representations and freeze the weights [Peters et al., 2019] during training after empirical trials[4]. Given the

[4] Using the representations of the last layer, or summing all the 12 layers' representations give consistently worse results.

sequence $\{x_1, x_2, ..., x_m\}$, we have:

$$\mathbf{xb}_1, \mathbf{xb}_2, ..., \mathbf{xb}_m = \texttt{BERT}(x_1, x_2, ..., x_m)$$

We forward the concatenation of the two representations for each token to the upper layers:

$$\mathbf{x}_i = \texttt{concat}(\mathbf{xe}_i, \mathbf{xb}_i)$$

BiLSTM Layer To help the model better capture task-specific features between the sequence tokens. We use a multi-layer (3 layers) bi-directional LSTM encoder on top of the token representations, which we denote as `BiLSTM`:

$$\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m\} = \texttt{BiLSTM}(\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m\})$$

CRF Layer Drawing inspirations for sentence-level sequence tagging models on tasks like NER [Lample et al., 2016]. Modeling the labeling decisions jointly rather than independently improves the models performance (e.g., the tag "I-Weapon" should not follow "B-Victim"). We model labeling decisions jointly using a conditional random field [Lafferty et al., 2001].

After passing $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_m\}$ through a linear layer, we have $\mathbf{P}$ of size $m\times$ size of tag space, where $\mathbf{P}_{i,j}$ is the score of the tag j of the i-th token in the sequence. For a tag sequence $\mathbf{y} = \{y_1, ..., y_m\}$, we have the score for the sequence-tag pair as:

$$\mathit{score}(\mathbf{X}, \mathbf{y}) = \sum_{i=0}^{m} \mathbf{A}_{y_i, y_{i+1}} + \sum_{i=1}^{m} \mathbf{P}_{i, y_i}$$

$\mathbf{A}$ is the *transition* matrix of scores such that $\mathbf{A}_{i,j}$ represents the score of a transition from the tag i to tag j. A `softmax` function is applied over scores for

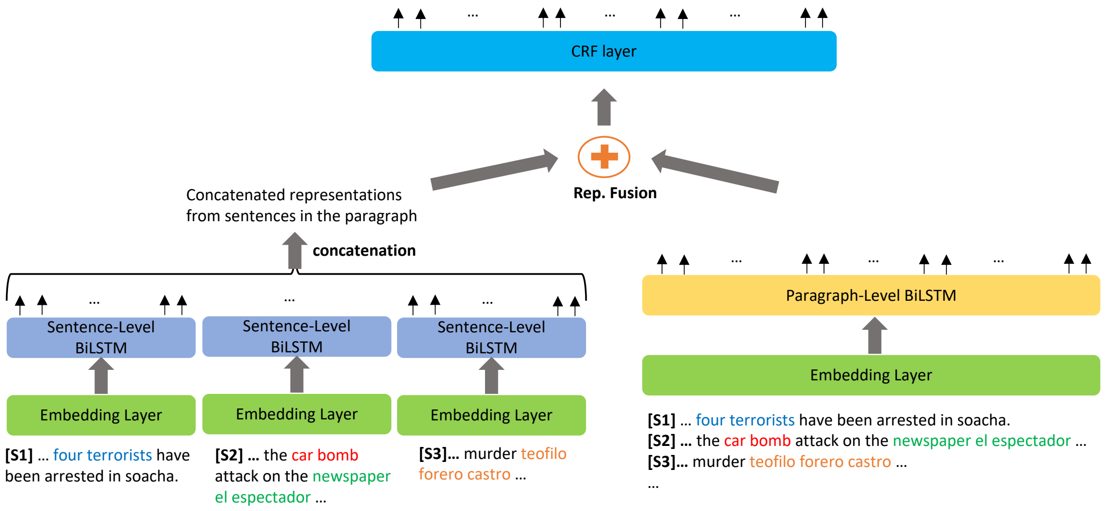

CRF layer
Rep. Fusion
Concatenated representations from sentences in the paragraph
concatenation
Sentence-Level BiLSTM
Sentence-Level BiLSTM
Sentence-Level BiLSTM
Embedding Layer
Embedding Layer
Embedding Layer
[S1] … four terrorists have been arrested in soacha.
[S2] … the car bomb attack on the newspaper el espectador …
[S3]… murder teofilo forero castro …
Paragraph-Level BiLSTM
Embedding Layer
[S1] … four terrorists have been arrested in soacha.
[S2] … the car bomb attack on the newspaper el espectador …
[S3]… murder teofilo forero castro …
…

Figure 3.3: Overview for our multi-granularity reader. The dark blue $\texttt{BiLSTM}_{sent.}$ produces sentence-level representations for each token, the yellow $\texttt{BiLSTM}_{para.}$ produces paragraph-level representations for each token.

all possible tag sequences, which yield a probability for the gold sequence $\mathbf{y}_{gold}$. The log-probability of the gold tag sequence is maximized during training. During decoding, the model predicts the output sequence that obtains the maximum score.

3.2.3 Multi-Granularity Reader

To explore the effect of aggregating contextualized token representations from different granularities (sentence- and paragraph-level), we propose the multi-granularity reader (Figure 3.3).

Similar to the general k-sentence reader, we use the same embedding layer here to represent the tokens. But we apply the embedding layer to two granularities of the paragraph text (sentence- and paragraph-level). Although the word embeddings are the same for the embedding layers from different granularities, the contextualized representations are different for each token – when the token is encoded in the context of a sentence, or in the context of a paragraph.

Correspondingly, we build two BiLSTMs ($\texttt{BiLSTM}_{sent.}$ and $\texttt{BiLSTM}_{para.}$) on top of the sentence-level contextualized token representations:

$$\{\tilde{\mathbf{x}}_1^{(1)}, ..., \tilde{\mathbf{x}}_{l_1}^{(1)}, ..., \tilde{\mathbf{x}}_{l_k}^{(k)}, ..., \tilde{\mathbf{x}}_{l_k}^{(k)}\}$$

,and the paragraph-level contextualized token representations:

$$\{\hat{\mathbf{x}}_1^{(1)}, ..., \hat{\mathbf{x}}_{l_1}^{(1)}, ..., \hat{\mathbf{x}}_{l_k}^{(k)}, ..., \hat{\mathbf{x}}_{l_k}^{(k)}\} :$$

Sentence-Level BiLSTM The $\texttt{BiLSTM}_{sent.}$ is applied sequentially to each sentence in the paragraph:

$$\{\tilde{\mathbf{p}}_1^{(1)}, \tilde{\mathbf{p}}_2^{(1)}, ..., \tilde{\mathbf{p}}_{l_1}^{(1)}\} = \texttt{BiLSTM}_{sent.}(\{\tilde{\mathbf{x}}_1^{(1)}, \tilde{\mathbf{x}}_2^{(1)}, ..., \tilde{\mathbf{x}}_{l_1}^{(1)}\})$$

$$...$$

$$\{\tilde{\mathbf{p}}_1^{(k)}, \tilde{\mathbf{p}}_2^{(k)}, ..., \tilde{\mathbf{p}}_{l_k}^{(k)}\} = \texttt{BiLSTM}_{sent.}(\{\tilde{\mathbf{x}}_1^{(k)}, \tilde{\mathbf{x}}_2^{(k)}, ..., \tilde{\mathbf{x}}_{l_k}^{(k)}\})$$

Then we have the sentence-level representations for each token in the paragraph as $\{\tilde{\mathbf{p}}_1^{(1)}, ..., \tilde{\mathbf{p}}_{l_1}^{(1)}, ..., \tilde{\mathbf{p}}_1^{(k)}, ..., \tilde{\mathbf{p}}_{l_k}^{(k)}\}$

Paragraph-Level BiLSTM Another BiLSTM layer ($\texttt{BiLSTM}_{para.}$) is applied to the *entire* paragraph (as compared to $\texttt{BiLSTM}_{sent.}$, which is applied to each sentence), to capture the dependency between tokens in the paragraph:

$$\{\hat{\mathbf{p}}_1^{(1)}, ..., \hat{\mathbf{p}}_{l_1}^{(1)}, ..., \hat{\mathbf{p}}_1^{(k)}, ..., \hat{\mathbf{p}}_{l_k}^{(k)}\} = \texttt{BiLSTM}_{para.}(\{\hat{\mathbf{x}}_1^{(1)}, ..., \hat{\mathbf{x}}_{l_1}^{(1)}, .., \hat{\mathbf{x}}_{l_k}^{(k)}, ..., \hat{\mathbf{x}}_{l_k}^{(k)}\})$$

Fusion and Inference Layer For each token $x_i^{(j)}$ (the i-th token in the j-th sentence), to fuse the representations learned at the sentence-level ($\tilde{\mathbf{p}}_i^{(j)}$) and paragraph-level ($\hat{\mathbf{p}}_i^{(j)}$), we propose two options – the first uses a sum operation, and the second uses a *gated* fusion operation:

- *Simple Sum Fusion*:

$$\mathbf{p}_i^{(j)} = \tilde{\mathbf{p}}_i^{(j)} + \hat{\mathbf{p}}_i^{(j)}$$

- *Gated Fusion*: The gated fusion compute the gate vector $\mathbf{g}_i^{(j)}$ with its sentence-level token representation $\tilde{\mathbf{p}}_i^{(j)}$ and paragraph-level token representation $\hat{\mathbf{p}}_i^{(j)}$, to control how much information should be incorporated from the two representations.

$$\mathbf{g}_i^{(j)} = sigmoid(\mathbf{W}_1\tilde{\mathbf{p}}_i^{(j)} + \mathbf{W}_2\hat{\mathbf{p}}_i^{(j)} + b)$$

$$\mathbf{p}_i^{(j)} = \mathbf{g}_i^{(j)} \odot \tilde{\mathbf{p}}_i^{(j)} + (1 - \mathbf{g}_i^{(j)}) \odot \hat{\mathbf{p}}_i^{(j)}$$

$$\odot : \text{element-wise product}$$

Similarly to in the general k-sentence reader, we add the CRF layer (section 3.2.2) on top of the fused representations for each token in the paragraph $\{\mathbf{p}_1^{(1)}, ..., \mathbf{p}_{l_1}^{(1)}, ..., \mathbf{p}_1^{(k)}, ..., \mathbf{p}_{l_k}^{(k)}\}$, to help jointly model the labeling decisions between tokens in the paragraph.

3.3 Experiments and Analysis

We evaluate our models' performance on the MUC-4 event extraction benchmark [MUC-4, 1992], and compare to prior work. We also report findings on the effect of context length on the end-to-end readers' performance on this document-level task.

3.3.1 Dataset and Evaluation Metrics

MUC-4 Event Extraction Dataset The MUC-4 dataset consists of 1,700 documents with associated answer key (role filler) templates. To make sure our results are comparable to the previously reported results on this dataset, we use the 1300 documents for training, 200 documents (`TST1+TST2`) as the development set and the 200 documents (`TST3+TST4`) as the test set.

Evaluation Metrics Following the prior work, we use *head noun phrase match* to compare the extractions against gold role fillers for evaluation [5]; besides noun phrase matching, we also report *exact match* accuracy to capture how well the models are capturing the role fillers' boundary[6]. Our results are reported as Precision (P), Recall (R) and F-measure (F-1) score for the macro average for all the event roles. In Table 3.2, we also present the scores for each event role (i.e., PERPETRATOR INDIVIDUALS, PERPETRATOR ORGANIZATIONS, PHYSICAL TARGETS, VICTIMS and WEAPONS) based on the head noun match metric.

3.3.2 Baseline Systems and Our Systems

We compare to the pipeline and manual feature engineering based systems: **GLACIER** [Patwardhan and Riloff, 2009] consists of a sentential event classifier and a set of plausible role filler recognizers for each event role. The final extraction decisions are based on the product of normalized sentential and phrasal probabilities; **TIER** [Huang and Riloff, 2011] proposes a multi-stage approach. It

[5] Duplicate role fillers (i.e., extractions for the same role that have the same head noun) are conflated before being scored; they are counted as one hit (if the system produces it) or one miss (if the system fails to produce any of the duplicate mentions).

[6] Similarly, duplicate extractions with the same string are counted as one hit or miss.

processes a document in three stages: classifying narrative document, recognizing event sentence and noun phrase analysis. **Cohesion Extract** [Huang and Riloff, 2012] adopts a bottom-up approach, which first aggressively identifies candidate role fillers in the document and then refines the candidate set with cohesion sentence classifier. Cohesion Extract obtains substantially better precision and with similar level of recall as compared to GLACIER and TIER.

To investigate how the neural models capture the long dependency in the context of variant length (single-sentence, double-sentence, paragraph or longer), we initialize the k in k-sentence reader to different values to build the: **Single-Sentence Reader** ($k = 1$), which reads through the document sentence-by-sentence to extract the event role fillers; **Double-Sentence Reader** ($k = 2$), which reads the document with step of two sentences; **Paragraph Reader** ($k = \#$ sentences in the paragraph), which reads the document paragraph-by-paragraph; **Chunk Reader** ($k =$ maximum # of sentences that fit right in the length constraint for pretrained LM models), which reads the document with the longest step (the constraint of BERT model).

The final row in Table 3.1 & 3.2 presents the results obtained with our **Multi-Granularity Reader**. Similar to the paragraph-level reader, it reads through document paragraph-by-paragraph, but learns the representations for both intra-sentence and inter-sentence context.

3.3.3 Results and Findings

We report the macro average results in Table 3.1. To understand in detail how the models extract the fillers for *each* event role, we also report the per event role

	Head Noun Match			Exact Match		
	Prec.	Recall	F-1	Prec.	Recall	F-1
GLACIER [Patwardhan and Riloff, 2009]	47.80	57.20	52.08	-	-	-
TIER [Huang and Riloff, 2011]	50.80	61.40	55.60	-	-	-
Cohesion Extract [Huang and Riloff, 2012]	57.80	59.40	58.59	-	-	-
w/o contextualized embedding						
Single-Sentence Reader	48.69	56.11	52.14	46.16	53.16	49.41
Double-sentence Reader	56.37	47.53	51.57	53.70	43.95	48.34
Paragraph Reader	53.19	53.16	53.17	49.45	49.26	49.35
Chunk Reader	**61.76**	37.04	46.31	**56.91**	34.92	43.28
w/ contextualized embedding						
Contextualized Single-Sentence Reader	47.32	61.26	53.39	44.40	**57.67**	50.17
Contextualized Double-sentence Reader	57.17	53.36	55.20	53.38	49.22	51.22
Contextualized Paragraph Reader	56.78	52.64	54.64	53.36	49.65	51.44
Contextualized Chunk Reader	60.90	41.10	49.07	55.18	37.51	44.66
Multi-Granularity Reader	56.44	**62.77**	**59.44**	52.03	56.81	**54.32**

Table 3.1: Macro average results for the document-level event role filler extraction task (highest number of the column boldfaced).

	PerpInd			PerpOrg			Target			Victim			Weapon		
	P	R	F-1	P	R	F-1	P	R	F-1	P	R	F-1	P	R	F-1
GLACIER [Patwardhan and Riloff, 2009]	51	58	54	34	45	38	42	72	53	55	58	56	57	53	55
TIER [Huang and Riloff, 2011]	54	57	56	55	49	51	55	68	61	63	59	61	62	64	63
Cohesion Extract [Huang and Riloff, 2012]	54	57	**56**	55	49	51	55	68	61	63	59	**61**	62	64	63
w/o contextualized embedding															
Single-Sentence Reader	38.38	50.68	43.68	40.98	69.05	51.44	62.50	42.76	50.78	36.69	55.79	44.27	64.91	62.30	63.58
Double-Sentence Reader	50.00	35.14	41.27	63.83	35.71	45.80	61.62	44.83	51.90	51.02	54.74	52.81	55.41	67.21	60.74
Paragraph Reader	42.51	51.35	46.52	44.80	54.76	49.28	70.33	43.45	53.71	53.75	47.37	50.36	54.55	68.85	60.87
Chunk Reader	65.63	26.19	37.44	50.00	45.45	47.62	77.78	22.62	35.05	55.00	21.15	30.56	60.42	69.77	64.76
w/ contextualized embedding															
C-Single-Sentence Reader	44.97	52.70	48.53	35.15	73.81	47.62	71.74	24.83	36.89	33.63	77.89	46.98	51.11	77.05	61.46
C-Double-Sentence Reader	63.49	31.76	42.34	53.25	48.81	50.93	69.52	50.34	58.40	44.03	62.11	51.53	55.56	73.77	63.38
C-Paragraph Reader	43.92	53.38	48.19	52.94	54.76	53.84	74.19	44.83	55.89	50.57	46.32	48.35	62.30	63.93	63.10
C-Chunk Reader	57.14	27.38	37.02	47.62	40.91	44.01	70.27	29.76	41.81	59.46	42.31	49.44	70.00	65.12	67.47
Multi-Granularity Reader	53.08	52.23	52.65	50.99	67.88	**58.23**	60.38	64.10	**62.18**	49.34	62.05	54.97	68.42	67.57	**67.99**

Table 3.2: Per event role results based on head noun match metric ("C-" stands for contextualized). The highest F-1 are boldfaced for each event role.

results in Table 3.2. We summarize the results into important findings below:

- *The end-to-end neural readers can achieve nearly the same level or significantly better results than the pipeline systems.* Although our models rely on no hand-designed features, the contextualized double-sentence reader and paragraph reader achieves nearly the same level of F-1 compared to Cohesion Extraction (CE), judging by the head noun matching metric. Our multi-granularity reader performs significantly better (∼60) than the prior state-of-the-art.

- *Contextualized embeddings for the sequence consistently improve the neural readers' performance.* The results show that the contextualized k-sentence readers all outperform their non-contextualized counterparts, especially when $k > 1$. The trends also exhibit in the per event role analysis (Table 3.2). To notice, we freeze the transformers' parameters during training (fine-tuning yields worse results).

- *It's not the case that modeling the longer context will result in better neural sequence tagging model on this document-level task.* When increasing the input context from a single sentence to two sentences, the reader has a better precision and lower recall, resulting in no better F-1; When increase the input context length further to the entire paragraph, the precision increases and recall remains the same level, resulting in higher F-1; When we keep increasing the length of input context, the reader becomes more conservative and F-1 drops significantly. All these indicate that focusing on the local (intra-sentence) and broader (paragraph-level) context are both important for the task. Similar results regarding the context length have also been found in document-level coreference resolution [Joshi et al., 2019].

- *Our multi-granularity reader that dynamically incorporates sentence-level and paragraph-level contextual information performs significantly better*, than the non end-to-end systems and our base k-sentence readers on the macro average F-1 metric. In terms of the per event role performance (Table 3.2), our reader: (1) substantially outperforms CE with a $\sim$ 7 F-1 gap on the Perpetrator Organization role; (2) slightly outperforms CE (∼1 on the Target category); (3) achieves nearly the same-level of F-1 on Perpetrator Individual and worse F-1 on Victim category.

3.4 Further Analysis

	Head Noun Match			Exact Match		
	Precision	Recall	F-1	Precision	Recall	F-1
Multi-granularity Reader	56.44	62.77	**59.44**	52.03	56.81	**54.32**
w/o gated fusion	48.09	67.32	56.10	43.75	62.37	51.43
w/o BERT	59.16	50.80	54.66	55.48	46.99	50.88
w/o CRF layer	50.52	56.95	53.54	47.02	53.55	50.07

Table 3.3: Ablation study on modules' influence on the multi-granularity reader.

We conduct an ablation study on how modules of our multi-granularity reader affect its performance on this document-level extraction task (Table 3.3). From the results, we find that: (1) when replacing the gated fusion operation with the simple sum of the sentence- and paragraph-level token representations, the precision and F-1 drop substantially, which proves the importance of dynamically incorporating context; (2) when removing the BERT's contextualized representations, the model becomes more conservative and yields substantially lower recall and F-1; (3) when replacing the CRF layer and make independent labeling decisions for each token, both the precision and recall drops substantially.

We also do an error analysis with examples and predictions from different models, to understand qualitatively the advantages and disadvantages of our models. In the first example below (green span: gold extraction, the role after is the span's event role), the multi-granularity (MG) reader and single-sentence reader correctly extracts the two target expressions, which the paragraph reader overlooks. Although only in the last sentence the attack and targets are mentioned, our MG reader successfully captures this with focusing on both the paragraph-level and intra-sentence context.

> ... the announcer says president virgilio barco will tonight disclose his government's peace proposal. .... Near the end, the announcer adds to the initial report on the el tomate attack with a 3-minute update that adds 2 injured, 21 houses Target destroyed, and 1 bus Target burned.

In the second example (red span: false positive perpInd extraction by the single-sentence reader), although "members of the civil group" appears in a sentence about explosion, judging from paragraph-level context or reasoning about the expression itself should help confirm that it is not perpetrator individual. The MG and paragraph reader correctly handles this and also extracts "the bomb".

> .... An attack came at approximately 22:30 last night. Members of the civil group and the peruvian investigative police went to the site of the explosion. The members of the republican guard antiexplosives brigade are investigating to determine the magnitude of the bomb Weapon used in this attack.

There's substantial improvement space for our MG reader's predictions. There are many role fillers which the reader overlooks. In the example below, "La Tandona" being a perpetrator organization is implicitly expressed in the document and the phrase did not appear elsewhere in the corpus. But external knowledge (e.g., Wikipedia) could help confirm its event role.

> ... Patriotic officer, it is time we sit down to talk, to see what we can do with our fatherland, and what are we going to do with La Tandona PerpOrg. .... To continue defending what, we ask you. ... .

In the last example, there are no explicit expression such as “kill” or “kidnap” in the context for the target. Thus it requires deeper understanding of the *entire* narrative and *reasoning* about the surrounding context to understand that “Jorge Serrano Gonzalez” is involved in a terrorism event.

> ... said that the guerrillas are desperate and ... . The president expressed his satisfaction at the release of Santander department senator Jorge Serrano Gonzalez Target, whom he described as one of the most important people that colombian democracy has at this moment.

3.5 Chapter Summary

In this chapter, we demonstrate that document-level event role filler extraction could be successfully tackled with end-to-end neural sequence models. Investigations on how the input context length affects the neural sequence readers’ performance show that context of very long length might be hard for the neural sequence labeling models to capture and results in lower performance. We propose a novel multi-granularity reader to dynamically incorporate not only sentence-level contextualized representations, but also paragraph-level representations. Evaluations on the benchmark dataset and qualitative analysis prove that our model achieves substantial improvement over prior work.

CHAPTER 4

GENERATIVE TRANSFORMERS FOR DOCUMENT-LEVEL EXTRACTION

In chapter 3, we discussed the importance of modeling contextual information beyond sentence-level and proposed the multi-granularity reader for extracting entity mentions. Apart from the additional context, modeling the structure and dependencies between different events is also important and is another challenge, as we've mentioned in chapter 4. In this chapter, we explore how to build neural generative models to model the within-event and across-event dependencies in the document-level template filling task. This work was published in Du et al. [2021a] and Du et al. [2021b].

Document-level template filling [Sundheim, 1991, 1993; Grishman and Sundheim, 1996] is a classic problem in information extraction (IE) and NLP [Jurafsky and Martin, 2014]. It is of great importance for automating many real-world tasks, such as event extraction from newswire [Sundheim, 1991]. The complete task is generally tackled in two steps. The first step detects events in the article and assigns templates to each of them (template recognition); the second step performs role-filler entity extraction (REE) for filling in the templates.

In contrast to sentence-level event extraction (see, e.g., the ACE evaluation [Linguistic Data Consortium, 2005]), document-level template filling introduces several complications especially in terms of the output structure. First, **role-filler entities must be extracted even if they never appear in the same sentence as an event trigger**. In Figure 4.1, for example, the WEAPON and the first mention of the telephone company building (TARGET) appear in a sentence that does not explicitly mention the explosion of the bomb; Second, **real documents**

often describe multiple events which are often related (Figure 4.6). From the example in Section 4.3, we can observe that between-events dependencies are important (e.g., a single organization can participate in multiple events) and can span the entire document (e.g., event-specific targets can be distant from their shared perpetrator organization). As a result of these complications, end-to-end sentence-level event extraction models [Chen et al., 2015; Lample et al., 2016], which dominate the literature, are ill-suited for the REE task, which calls for models that encode information and track entities across a longer context.

Fortunately, neural models for event extraction that have the ability to model longer contexts have been developed. Du and Cardie [2020], for example, extend standard contextualized representations [Devlin et al., 2019] to produce a document-level sequence tagging model for event argument extraction. Both approaches show improvements in performance over sentence-level models on event extraction. Regrettably, these approaches (as well as most sentence-level methods) handle each candidate role-filler prediction and event/template detection in isolation. Consequently, they cannot easily exploit *semantic dependencies between closely related roles* like the PerpInd and the PerpOrg, which can share a portion of the same entity span. "Shining Path members", for instance, describes the PerpInd in Figure 4.1, and its sub-phrase, "Shining Path", describes the associated PerpOrg. In addition, prior models cannot model the dependencies between multiple templates across the template (e.g., some types of events are more likely to co-occur).

Motivated by these, we introduce a novel end-to-end generative transformer model — the "**G**enerative **R**ole-f**i**ller **T**ransformer" (Grit) (Section 4.2) for the REE sub-task. Then we extend the Grit model to build our framework Gtt (Section 4.3), and investigate the generative transformers' potential in tackling the entire template filling task in an end-to-end manner, instead of doing REE and event recognition with two systems.

In Section 4.2 we focus on the role-filler entity extraction (REE) sub-task of template filling (Figure 4.1)[1] and modeling details for our GRIT model. The input text describes a bombing event; the goal is to identify the entities that fill any of the roles associated with the event (e.g., the perpetrator, their organization, the weapon) by extracting a descriptive "mention" of it – a string from the document. In Section 4.3, we extend our GRIT model and propose the first end-to-end generative learning framework (GTT) for the template filling full task.

For experiments, we evaluate GRIT on the MUC-4 [MUC-4, 1992] REE task. Empirically, our model outperforms substantially strong baseline models. We also demonstrate that GRIT is better than existing document-level event extraction approaches at capturing linguistic properties critical for the task, including coreference between entity mentions and cross-role extraction dependencies. We evaluate GTT on the MUC-4 template filling task. Empirically, our model substantially outperforms both pipeline-based and end-to-end baseline models. In our analysis, we demonstrate that our model is better at capturing between-event dependencies, which are critical for documents that describe multiple events.

4.1 Related Work

Sentence-level Event Extraction Most work in event extraction has focused on the ACE sentence-level event task [Walker et al., 2006], which requires the

[1] In this sub-task, we assume there is one generic template for the entire document [Huang and Riloff, 2011, 2012].

detection of an event trigger and extraction of its arguments from within a single sentence. Previous state-of-the-art methods include Li et al. [2013] and Li et al. [2015], which explored a variety of hand-designed features. More recently, neural network based models such as recurrent neural networks [Nguyen et al., 2016; Feng et al., 2018], convolutional neural networks [Nguyen and Grishman, 2015; Chen et al., 2015] and attention mechanisms [Liu et al., 2017, 2018] have also been shown to help improve performance. Beyond the task-specific features learned by the deep neural models, Zhang et al. [2019b] and Wadden et al. [2019] also utilize pre-trained contextualized representations.

Only a few models have gone beyond individual sentences to make decisions. Ji and Grishman [2008] and Liao and Grishman [2010] utilize event type co-occurrence patterns to propagate event classification decisions. Yang and Mitchell [2016] propose to learn within-event (sentence) structures for jointly extracting events and entities within a document context. Similarly, from a methodological perspective, our Grit and Gtt model also learn structured information, but it learns the dependencies between different roles (within one template), as well between multiple events (across multiple templates). Duan et al. [2017b] and Zhao et al. [2018] leverage document embeddings as additional features to aid event detection. Although the approaches above make decisions with cross-sentence information, their extractions are still done the sentence level.

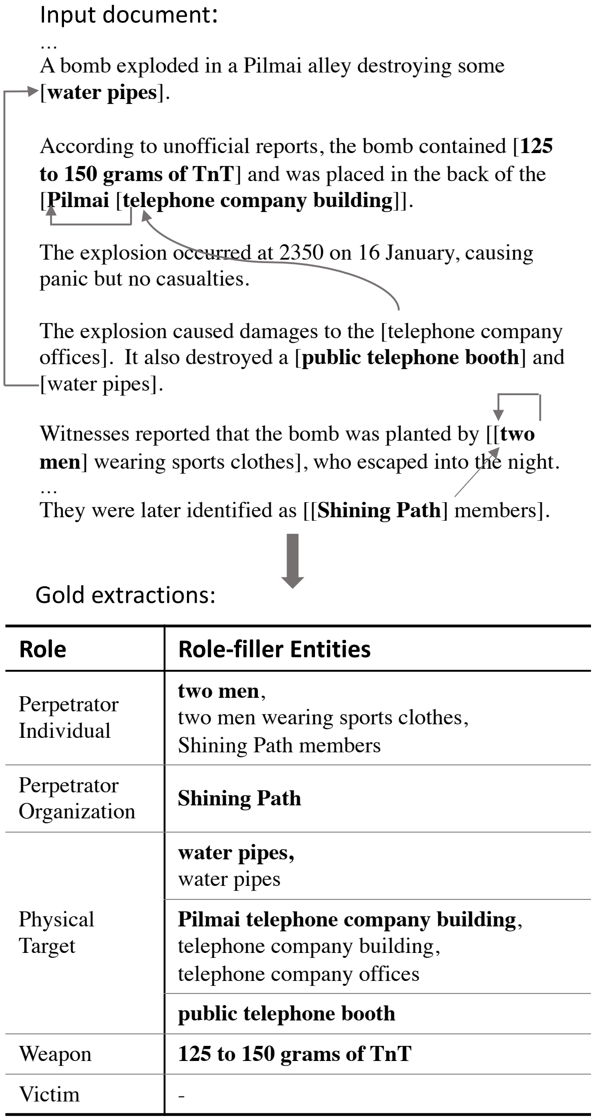
Document-level IE Document-level event role-filler *mention* extraction has been explored in recent work, using hand-designed features for both local and additional context [Patwardhan and Riloff, 2009; Huang and Riloff, 2011, 2012], and with end-to-end sequence tagging based models with contextualized pre-trained representations [Du and Cardie, 2020]. These efforts are the most related to our work. The key difference is that our work focuses on a more challenging, and more realistic, setting: extracting role-filler entities rather than lists of role-filler mentions that are not grouped according to their associated entity. Also on a related note, Chambers and Jurafsky [2011], Chambers [2013], and Liu et al. [2019] work on *unsupervised* event schema induction and open-domain event extraction from documents.

The full task of event template filling consists of REE and then grouping extracted entities into different templates (each of them representing an event). Simplifications of the task Patwardhan and Riloff [2009]; Huang and Riloff [2011, 2012]; Du et al. [2021a] assume that there is one generic template and focus only on role-filler entity extraction. However, real documents often describe multiple events (Figure 4.6). Thus we investigate how to adapt of model for REE, for the full task of template filling, which is more challenging.

Neural Generative Models with a Shared Module for Encoder and Decoder Our Grit model uses one shared transformer module for both the encoder and decoder, which is simple and effective. For the machine translation task, He et al. [2018] propose a model which shares the parameters of each layer between the encoder and decoder to regularize and coordinate the learning. Dong et al. [2019] presents a new unified pre-trained language model that can be fine-tuned for both NLU and NLG tasks.

4.2 Extracting Role-filler Entity (GRIT)

4.2.1 The Role-filler Entity Extraction Task and Evaluation Metric



Input document:

...
A bomb exploded in a Pilmai alley destroying some [**water pipes**].

According to unofficial reports, the bomb contained [**125 to 150 grams of TnT**] and was placed in the back of the [**Pilmai** [**telephone company building**]].

The explosion occurred at 2350 on 16 January, causing panic but no casualties.

The explosion caused damages to the [telephone company offices]. It also destroyed a [**public telephone booth**] and [water pipes].

Witnesses reported that the bomb was planted by [[**two men**] wearing sports clothes], who escaped into the night.
...
They were later identified as [[**Shining Path**] members].

Gold extractions:

Role	Role-filler Entities
Perpetrator Individual	**two men**, two men wearing sports clothes, Shining Path members
Perpetrator Organization	**Shining Path**
Physical Target	**water pipes,** water pipes
	Pilmai telephone company building, telephone company building, telephone company offices
	public telephone booth
Weapon	**125 to 150 grams of TnT**
Victim	-

Figure 4.1: Role-filler entity extraction (REE). The first mention of each role-filler entity is bold in the table and document. The arrows denote coreferent mentions.

We base the REE task on the original MUC[2] formulation [Sundheim, 1991], but simplify it as done in prior research [Huang and Riloff, 2012; Du and Cardie, 2020]. In particular, we assume that one generic template should be produced for each document: for documents that recount more than one event, the extracted role-filler entities for each are merged into a single event template. Second, we focus on entity-based roles with string-based fillers[3]. The REE task is illustrated by an example in 4.1.

- Each *event* consists of the set of roles that describe it (shown in Figure 4.1). The MUC-4 dataset that we use consists of ∼1k terrorism events.
- Each *role* is filled with one or more entities. There are five such roles for MUC-4: perpetrator individuals (PerpInd), perpetrator organizations (PerpOrg), physical targets (Target), victims (Victim) and weapons (Weapon). These event roles represent the agents, patients, and instruments associated with terrorism events [Huang and Riloff, 2012].
- Each *role-filler entity* is denoted by a single descriptive *mention*, a span of text from the input document. Because multiple such mentions for each entity may appear in the input, the gold-standard template lists all alternatives (shown in Figure 4.1), but systems are required to produce just one.

Evaluation Metric The metric for past work on document-level role-filler *mentions* extraction [Patwardhan and Riloff, 2009; Huang and Riloff, 2011; Du and Cardie, 2020] calculates mention-level precision across all alternative mentions for

[2] The Message Understanding Conferences were a series of U.S. government-organized IE evaluations.

[3] Other types of role fillers include normalized dates and times, and categorical "set" fills. We do not attempt to handle these in the current work.

each role-filler entity. Thus it is not suited for our problem setting, where entity-level precision is needed, where spurious entity extractions will get punished (e.g., recognizing "telephone company building" and "telephone company offices" as two entities will result in lower precision).

Drawing insights from the entity-based CEAF metric [Luo, 2005] from the coreference resolution literature, we design a metric (**CEAF-REE**) for measuring models' performance on this document-level **role-filler entity extraction** task. It is based on maximum bipartite matching algorithm [Kuhn, 1955; Munkres, 1957]. The general idea is that, for each role, the metric is computed by aligning gold and predicted entities with the constraint that a predicted (gold) entity is aligned with at most one gold (predicted) entity. Thus, the system that does not recognize the coreferent mentions and use them for separate entities will be penalized in precision score. For the example in Figure 4.1, if the system extracts "Pilmai telephone company building" and "telephone company offices" as two distinct Targets, the precision will drop. We include more details for our CEAF-TF metric in the appendix.

4.2.2 REE as Sequence Generation

We treat document-level REE as a sequence-to-sequence task [Sutskever et al., 2014] in order to better model the cross-role dependencies and cross-sentence noun phrase coreference structure. We first transform the task definition into a source and target sequence.

As shown in Figure 4.2, the *source sequence* simply consists of the tokens of the original document prepended with a "classification" token (i.e., [CLS] in BERT), and appended with a separator token (i.e., [SEP] in BERT). The *target sequence* is the concatenation of target extractions for each role, separated by the separator token. For each role, the target extraction consists of the first mention's beginning (b) and end (e) tokens:

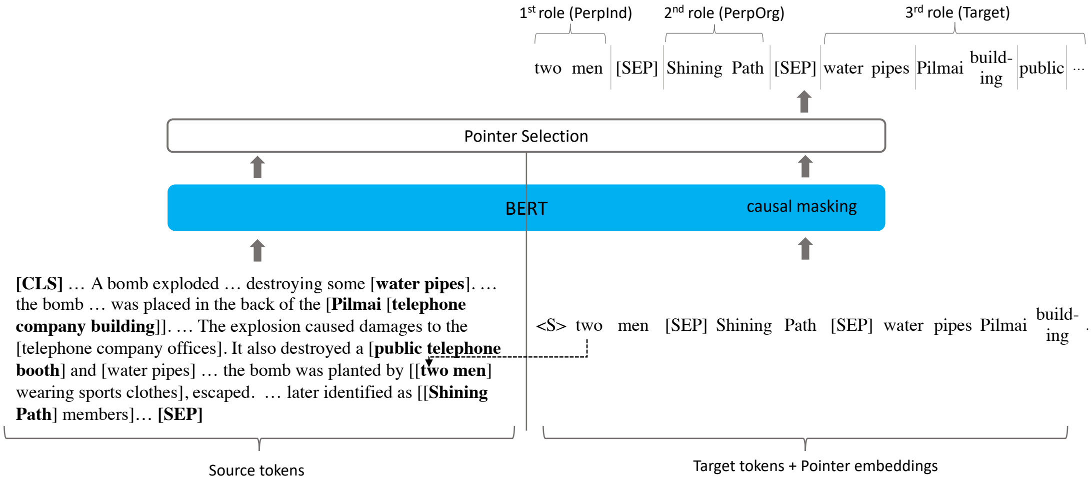
$$\text{<S>}\ e^{(1)}_{1_b}, e^{(1)}_{1_e}, \ldots\ \text{[SEP]}$$

$$e^{(2)}_{1_b}, e^{(2)}_{1_e}, \ldots\ \text{[SEP]}$$

$$e^{(3)}_{1_b}, e^{(3)}_{1_e}, e^{(3)}_{2_b}, e^{(3)}_{2_e}, \ldots\ \text{[SEP]}$$

$$\ldots$$

Note that we list the roles in a fixed order for all examples. So for the example used in Figure 4.2, $e^{(1)}_{1_b}$, $e^{(1)}_{1_e}$ would be "two" and "men" respectively; and $e^{(3)}_{1_b}$, $e^{(3)}_{1_e}$ would be "water" and "pipes" respectively. Henceforth, we denote the resulting sequence of source tokens as $x_0, x_1, ..., x_m$ and the sequence of target tokens as $y_0, y_1, ..., y_n$.


1st role (PerpInd)
2nd role (PerpOrg)
3rd role (Target)
two men [SEP] Shining Path [SEP] water pipes Pilmai build-ing public ...
Pointer Selection
BERT
causal masking
<S> two men [SEP] Shining Path [SEP] water pipes Pilmai build-ing ...
Source tokens
Target tokens + Pointer embeddings

[CLS] … A bomb exploded … destroying some [**water pipes**]. … the bomb … was placed in the back of the [**Pilmai** [**telephone company building**]]. … The explosion caused damages to the [telephone company offices]. It also destroyed a [**public telephone booth**] and [water pipes] … the bomb was planted by [[**two men**] wearing sports clothes], escaped. … later identified as [[**Shining Path**] members]… **[SEP]**

Figure 4.2: Grit: generative transformer model for document-level event role-filler entity extraction. (Noun phrase bracketing and **bold** in the source tokens are provided for readability purposes and are not part of the source sequence.)

4.2.3 Model: Generative Role-filler Transformer

Our model is shown in Figure 4.2. It consists of two parts: the encoder (left) for the source tokens; and the decoder (right) for the target tokens. Instead of using a sequence-to-sequence learning architecture with separate modules [Sutskever et al., 2014; Bahdanau et al., 2015], we use a single pretrained transformer model [Devlin et al., 2019] for both parts, and introduce no additional fine-tuned parameters.

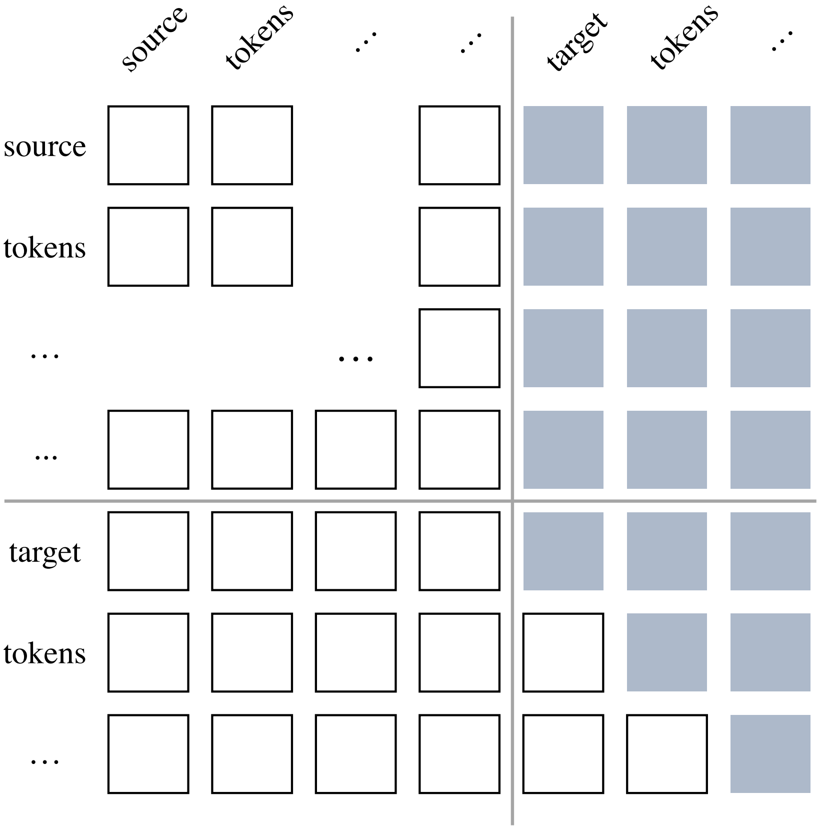


Figure 4.3: Partially causal masking strategy (**M**). (White cell: unmasked; Grey cell: masked).

Pointer Embeddings The first change to the model is to ensure that the decoder is aware of where its previous predictions come from in the source document, an approach we call "pointer embeddings". Similar to BERT, the input to the model consists of the sum of token, position and segment embeddings. However, for the position we use the corresponding source token's position. For example, for the word "two", the target tokens would have the identical position embedding of the word "two" in the source document. Interestingly, we do not use any explicit target position embeddings, but instead separate each role with a [SEP] token. Empirically, we find that the model is able to use these separators to learn which role to fill and which mentions have filled previous roles.

Our encoder's embedding layer uses standard BERT embedding layer, which applied to the source document tokens. To denote boundary between source and target tokens, we use sequence A (first sequence) segment embeddings for the source tokens, we use sequence B (second sequence) segment embeddings for the target tokens.

We pass the source document tokens through the encoder's embedding layer, to obtain their embeddings $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_m$. We pass the target tokens $y_0, y_1, ..., y_n$ through the decoder's embedding layer, to obtain $\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_n$.

BERT as Encoder / Decoder We utilize one BERT model as both the source and target embeddings. To distinguish the encoder / decoder representations, we provide a partial causal attention mask on the decoder side.

In Figure 4.3, we provide an illustration for the attention masks – 2-dimensional matrix denoted as $\mathbf{m}$. For the source tokens, the mask allows full source self-attention, but mask out all target tokens. For $i \in \{0, 1, ..., m\}$,

$$\mathbf{M}_{i,j} = \begin{cases} 1, & \text{if } \ 0 \leq j \leq m \\ 0, & \text{otherwise} \end{cases}$$

For the target tokens, to guarantee that the decoder is autoregressive (the current token should not attend to future tokens), we use a causal masking strategy. Assuming we concatenate the target to the source tokens (the joint sequence mentioned below), for $i \in \{m+1, ..., n\}$,

$$\mathbf{M}_{i,j} = \begin{cases} 1, & \text{if } \ 0 \leq j \leq m \\ 1, & \text{if } \ j > m \ \text{ and } \ j \leq i \\ 0, & \text{otherwise} \end{cases}$$

The joint sequence of source tokens' embeddings $(\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_m)$ and target tokens' embeddings $(\mathbf{y}_0, \mathbf{y}_1, ..., \mathbf{y}_n)$ are passed through BERT to obtain their contextualized representations,

$$\begin{aligned} &\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_m, \hat{\mathbf{y}}_0 ..., \hat{\mathbf{y}}_n \\ &= \texttt{BERT}(\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_m, \mathbf{y}_0, ..., \mathbf{y}_n) \end{aligned}$$

Pointer Decoding For the final layer, we replace word prediction with a simple pointer selection mechanism. For target time step t $(0 \leq t \leq n)$, we first calculate the dot-product between $\hat{\mathbf{y}}_t$ and $\hat{\mathbf{x}}_0, \hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_m$,

$$z_0, z_1, ..., z_m = \hat{\mathbf{y}}_t \cdot \hat{\mathbf{x}}_0, \hat{\mathbf{y}}_t \cdot \hat{\mathbf{x}}_1, ..., \hat{\mathbf{y}}_t \cdot \hat{\mathbf{x}}_m$$

Then we apply softmax to $z_0, z_1, ..., z_m$ to obtain the probabilities of pointing to each source token,

$$p_0, p_1, ..., p_m = \text{softmax}(z_0, z_1, ..., z_m)$$

Test prediction is done with greedy decoding. At each time step t, argmax is applied to find the source token which has the highest probability. The predicted token is added to the target sequence for the next time step $t+1$ with its pointer embedding. We stop decoding when the fifth [SEP] token is predicted, which represents the end of extractions for the last role.

In addition, we add the following decoding constraints,

- Tune probability of generating [SEP]. By doing this, we encourage the model to point to other source tokens and thus extract more entities for each role, which will help increase the recall. (We set the hyperparameter of downweigh to 0.01, i.e., for the [SEP] token $p = 0.01 * p$.)
- Ensure that the token position increase from start token to end token. When decoding tokens for each role, we know that mention spans should obey this property. Thus we eliminate those invalid choices during decoding.

4.2.4 Experimental Setup

We conduct evaluations on the MUC-4 dataset MUC-4 [1992], and compare to recent competitive end-to-end models [Wadden et al., 2019; Du and Cardie, 2020] in IE (Section 4.2.5). Besides the normal evaluation, we are also interested in how well our GRIT model captures coreference linguistic knowledge, and comparison with the prior models. In Section 4.2.6, we present relevant evaluations on the subset of test documents.

Dataset and Evaluation Metric The MUC-4 dataset consists of 1,700 documents with associated templates. Similar to [Huang and Riloff, 2012; Du and Cardie, 2020], we use the 1300 documents for training, 200 documents (`TST1+TST2`) as the development set and 200 documents (`TST3+TST4`) as the test set. Each document in the dataset contains on average 403.27 tokens, 7.12 paragraphs. In Table 4.1, we include descriptions for each role in the template.

Roles	Descriptions
PerpInd	A person responsible for the incident.
PerpOrg	An organization responsible for the incident.
Target	A thing (inanimate object) that was attacked.
Victim	The name of a person who was the obvious or apparent target of the attack or who became a victim of the attack.
Weapon	A device used by the perpetrator(s) in carrying.

Table 4.1: Natural Language Descriptions for Each Role.

We use the first appearing mention of the role-filler entity as the training signal (thus do not use the other alternative mentions during training).

We use CEAF-REE which is covered in Section 4.2.1 as the evaluation metric. The results are reported as Precision (P), Recall (R) and F-measure (F1) score for the micro-average for all the event roles (Table 4.2). We also report the per-role results to have a fine-grained understanding of the numbers (Table 4.3).

Baselines We compare to recent strong models for (document-level) information/event extraction. CohesionExtract [Huang and Riloff, 2012] is a bottom-up approach for event extraction that first aggressively identifies candidate role-fillers, and prune the candidates located in event-irrelevant sentences.[4] Du and Cardie [2020] propose *neural sequence tagging (NST)* models with contextualized representations for document-level role filler mentions extraction. We train this model with BIO tagging scheme to identify the first mention for each role-filler entity and its type (i.e., B-PerpInd, I-PerpInd for perpetrator individual). DyGIE++ [Wadden et al., 2019] is a span-enumeration based extraction model for entity, relation,

[4] Instead of using feature-engineering based sentence classification to identify event-relevant sentences, we re-implement the sentence classifier with BiLSTM-based neural sequence model.

and event extraction. The model (1) enumerates all the possible spans in the document; (2) concatenates the representations of the span's beginning & end token and use it as its representation, and pass it through a classifier layer to predict whether the span represents certain role-filler entity and what the role is. Both the NST and DyGIE++ are end-to-end and fine-tuned BERT [Devlin et al., 2019] contextualized representations with task-specific data. We train them to identify the first mention for each role-filler entity (to ensure fair comparison with our proposed model). Unsupervised event schema induction based approaches [Chambers and Jurafsky, 2011; Chambers, 2013; Cheung et al., 2013] are also able to model the coreference relations and entities at document-level, but have been proved to perform substantially worse than supervised models [Patwardhan and Riloff, 2009; Huang and Riloff, 2012]. Thus we do not compare with them. We also experimented with a variant of our Grit model – instead of always pointing to the same [SEP] in the source tokens to finish extracting the role-filler entities for a role, we use five different [SEP] tokens. During decoding, the model points to the corresponding [SEP] as the end of extraction for that role. This variant does not improve over the current best results and we omit reporting its performance.

4.2.5 Results

In Table 4.2, we report the micro-average performance on the test set. We observe that our Grit model substantially outperforms the baseline extraction models in precision and F1, with an over 5% improvement in precision over DyGIE++.

Table 4.3 compares the models' performance scores on each role (PerpInd, PerpOrg, Target, Victim, Weapon). We see that, (1) our model achieves the best precision across the roles; (2) for the roles that come with entities con-

Models	P	R	F1
CohesionExtract [Huang and Riloff, 2012]	58.38	39.53	47.14
NST [Du and Cardie, 2020]	56.82	**48.92**	52.58
DyGIE++ [Wadden et al., 2019]	57.04	46.77	51.40
Grit	**64.19****	47.36	**54.50***

Table 4.2: Micro-average results (the highest number of each column is boldfaced). Significance is indicated with $^{**}(p < 0.01)$,$^{*}(p < 0.1)$ – all tests are computed using the paired bootstrap procedure [Berg-Kirkpatrick et al., 2012].

	PerpInd	PerpOrg	Target	Victim	Weapon
NST [Du and Cardie, 2020]	48.39 / 32.61 / 38.96	60.00 / 43.90 / 50.70	54.96 / 52.94 / **53.93**	62.50 / 63.16 / 62.83	61.67 / 61.67 / **61.67**
DyGIE++ [Wadden et al., 2019]	59.49 / 34.06 / 43.32	56.00 / 34.15 / 42.42	53.49 / 50.74 / 52.08	60.00 / 66.32 / 63.00	57.14 / 53.33 / 55.17
Grit	65.48 / 39.86 / **49.55**	66.04 / 42.68 / **51.85**	55.05 / 44.12 / 48.98	76.32 / 61.05 / **67.84**	61.82 / 56.67 / 59.13

Table 4.3: Per-role performance scored by CEAF-REE (reported as P/R/F1, highest F1 for each role are boldfaced).

taining more human names (e.g., PerpInd and Victim), our model substantially outperforms the baselines; (3) for the role PerpOrg, our model scores better precision but lower recall than neural sequence tagging, which results in a slightly better F1 score; (4) for the roles Target and Weapon, our model is more conservative (lower recall) and achieves lower F1. One possibility is that for role like Target, on average there are more entities (though with only one mention each), and it's harder for our model to decode as many Target entities correct in a generative way.

4.2.6 Discussion

How well do the models capture coreference relations between mentions? We also conduct targeted evaluations on subsets of test documents whose

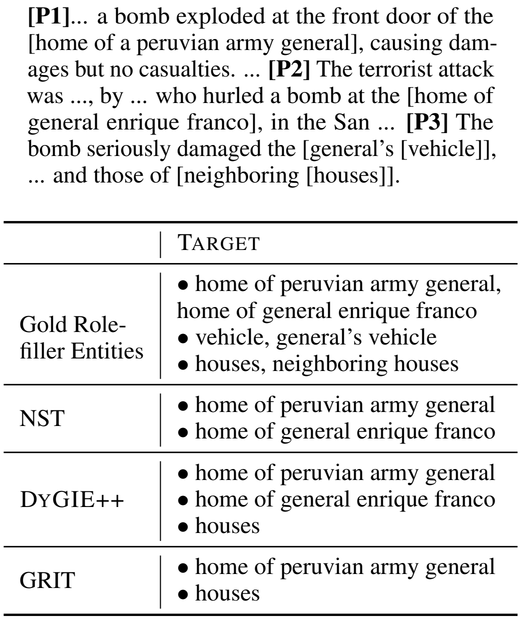
	$k = 1$	$1 < k \leq 1.25$	$1.25 < k \leq 1.5$	$1.5 < k \leq 1.75$	$k > 1.75$
NST [Du and Cardie, 2020]	63.83 / 51.72 / 57.14	57.45 / 38.57 / 46.15	60.32 / 49.03 / 54.09	64.81 / 50.00 / 56.45	66.67 / 51.90 / 58.36
DyGIE++ [Wadden et al., 2019]	**72.50** / 50.00 / 59.18	70.00 / 40.00 / 50.91	60.48 / 48.39 / 53.76	52.94 / 38.57 / 44.63	66.96 / 48.73 / 56.41
Grit	65.85 / 46.55 / 54.55	**74.42** / 45.71 / 56.64	**73.20** / 45.81 / 56.35	**67.44** / 41.43 / 51.33	**69.75** / 52.53 / 59.93

Table 4.4: Evaluations on the subsets of documents with increasing number of mentions per role-filler entity. k denotes the average # mentions per role-filler entity. Results for each column are reported as Precision / Recall / F1. The highest precisions are boldfaced for each bucket.

gold extractions come with coreferent mentions. From left to right in Table 4.4, we report results on the subsets of documents with increasing number (k) of possible (coreferent) mentions per role-filler entity. We find that: (1) On the subset of documents with only one mention for each role-filler entity ($k = 1$), our model has no significant advantage over DyGIE++ and the sequence tagging based model; (2) But as k increases, the advantage of our GRIT substantially increases – with an over 10% gap in precision when $1 < k \leq 1.5$, and a near 5% gap in precision when $k > 1.5$.

From the qualitative example (document excerpt and the extractions in Figure 4.4), we also observe our model recognizes the coreference relation between candidate role-filler entity mentions, while the baselines do not, which shows that our model is better at capturing the (non-)coreference relations between role-filler entity mentions. It also proves the advantage of a generative model in this setting.

How well do models capture dependencies between different roles? To study this phenomenon, we consider nested role-filler entity mentions in the documents. In the example of Figure 4.1, "shining path" is a role-filler entity mention for PerpOrg nested in "two *shining path* members" (a role-filler entity mention for PerpInd). The nesting happens more often between more related roles (e.g., PerpInd and PerpOrg) – we find that 33 out of the 200 test documents' gold



[P1]... a bomb exploded at the front door of the [home of a peruvian army general], causing damages but no casualties. ... **[P2]** The terrorist attack was ..., by ... who hurled a bomb at the [home of general enrique franco], in the San ... **[P3]** The bomb seriously damaged the [general's [vehicle]], ... and those of [neighboring [houses]].

	TARGET
Gold Role-filler Entities	• home of peruvian army general, home of general enrique franco • vehicle, general's vehicle • houses, neighboring houses
NST	• home of peruvian army general • home of general enrique franco
DYGIE++	• home of peruvian army general • home of general enrique franco • houses
GRIT	• home of peruvian army general • houses

Figure 4.4: Our model implicitly captures coreference relations between mentions.

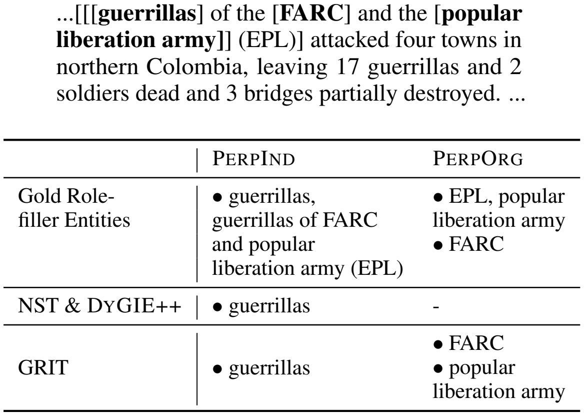
extractions contain nested role-filler entity mentions between the two roles.

	PERPORG (all docs)	PERPORG (33/200)
	P / R / F1	P / R / F1
NST	56.00 / 34.15 / 42.42	80.00 / 44.44 / 57.14
DYGIE++	60.00 / **43.90** / 50.70	61.54 / 35.56 / 45.07
GRIT	66.04 / 42.68 / 51.85	80.77 / **46.67** / 59.15

Table 4.5: Evaluation on the subset of documents that have nested role-filler entity mentions between role PERPIND and PERPORG (highest recalls boldfaced).

In Table 4.5, we present the CEAF-REE scores for role PERPORG on the subset of documents with nested roles. As we hypothesized beforehand, GRIT is able to learn the dependency between different roles and can learn to avoid missing relevant role-filler entities for later roles. The results provide empirical evidence: by learning the dependency between PERPIND and PERPORG, GRIT improves the relative recall score on the subset of documents as compared to DYGIE++. On all the 200 test documents, our model is $\sim$ 2% below DYGIE++ in recall;

while on the 33 docs, our model scores much higher than DyGIE++ in recall.



...[[[**guerrillas**] of the [**FARC**] and the [**popular liberation army**]] (EPL)] attacked four towns in northern Colombia, leaving 17 guerrillas and 2 soldiers dead and 3 bridges partially destroyed. ...

	PerpInd	PerpOrg
Gold Role-filler Entities	• guerrillas, guerrillas of FARC and popular liberation army (EPL)	• EPL, popular liberation army • FARC
NST & DyGIE++	• guerrillas	-
GRIT	• guerrillas	• FARC • popular liberation army

Figure 4.5: Our model captures dependencies between different roles.

For the document in the example of Figure 4.5, our model correctly extracts the two role-filler entities for PerpOrg: “FARC” and “popular liberation army”, which are closely related to the PerpInd entity “guerrilla”. While DyGIE++ and NST both miss the entities for PerpOrg.

Decoding Ablation Study In the table below, we present ablation results based on the decoding constraints. These illustrate the influence of the decoding constraints on the our model’s performance. The two constraints both significantly improve model predictions. Without downweighing the probability of pointing to [SEP], the precision increases but recall and F1 significantly drops.

	P	R	F1	Δ (F1)
Grit	64.19	47.36	54.50	
− [SEP] downweigh	67.43	40.12	50.31	-4.19
− constraint on pointer offset	62.90	45.79	53.00	-1.50

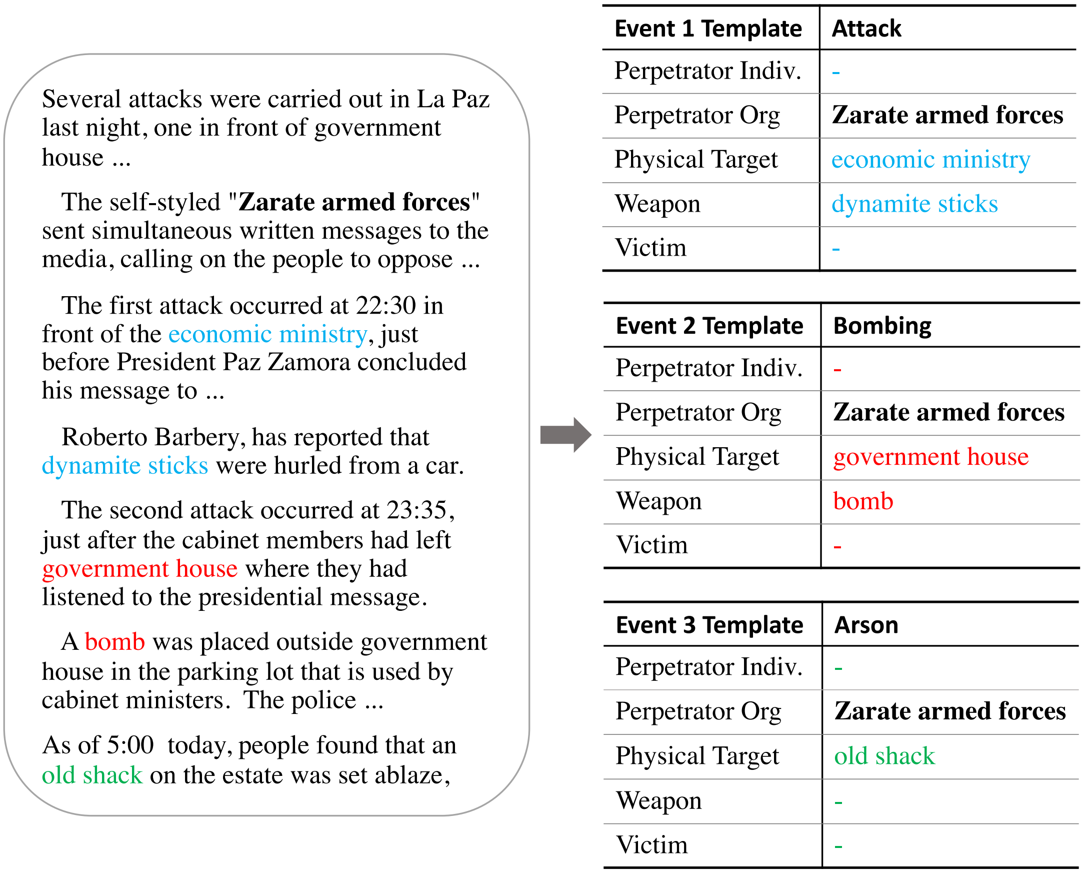
Table 4.6: Decoding Ablation Study

Additional Parameters and Training Cost Finally we consider additional parameters and training time of the models: As we introduced previously, the baseline models DyGIE++ and NST both require an additional classifier layer on top of BERT's hidden state (of size H) for making the predictions. While our Grit model does not require adding any new parameters. As for the training time, training the DyGIE++ model takes over 10 times longer time than NST and our model. This time comes from the DyGIE++ model requirement of enumerating all possible spans (to a certain length constraint) in the document and calculating the loss with their labels.

	additional params	training cost
DyGIE++	$2H(\#\text{roles} + 1)$	~20h
NST	$H(2\#\text{roles} + 1)$	~1h
Grit	0	<40min

Table 4.7: Additional Parameters and Training Cost.

4.3 Extracting Event Templates (GTT)



Several attacks were carried out in La Paz last night, one in front of government house ...

The self-styled "**Zarate armed forces**" sent simultaneous written messages to the media, calling on the people to oppose ...

The first attack occurred at 22:30 in front of the economic ministry, just before President Paz Zamora concluded his message to ...

Roberto Barbery, has reported that dynamite sticks were hurled from a car.

The second attack occurred at 23:35, just after the cabinet members had left government house where they had listened to the presidential message.

A bomb was placed outside government house in the parking lot that is used by cabinet ministers. The police ...

As of 5:00 today, people found that an old shack on the estate was set ablaze,

Event 1 Template	Attack
Perpetrator Indiv.	-
Perpetrator Org	**Zarate armed forces**
Physical Target	economic ministry
Weapon	dynamite sticks
Victim	-

Event 2 Template	Bombing
Perpetrator Indiv.	-
Perpetrator Org	**Zarate armed forces**
Physical Target	government house
Weapon	bomb
Victim	-

Event 3 Template	Arson
Perpetrator Indiv.	-
Perpetrator Org	**Zarate armed forces**
Physical Target	old shack
Weapon	-
Victim	-

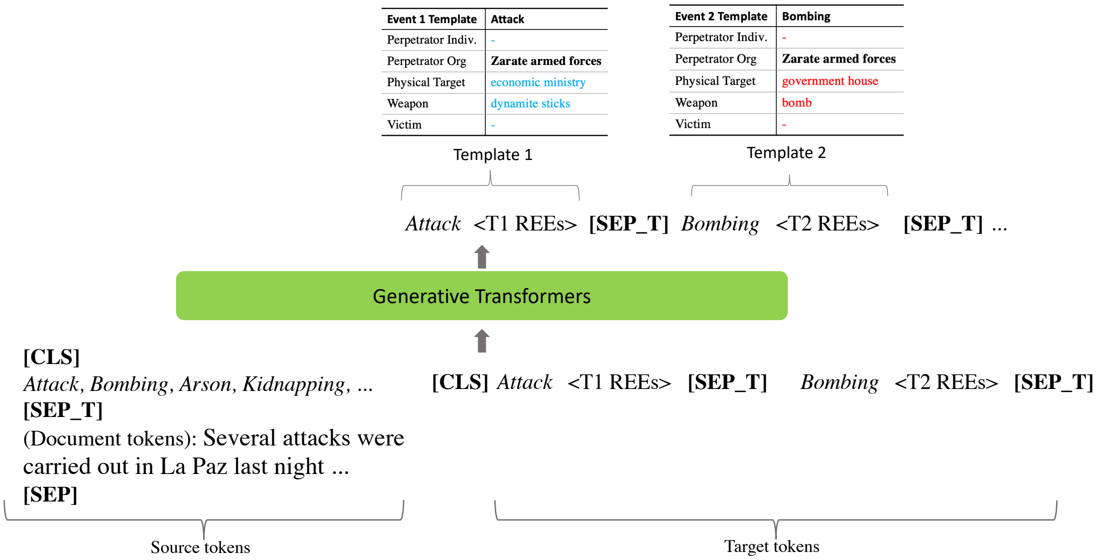
Figure 4.6: The template-filling task. Role-filler entity extraction is shown on the left, and template recognition is shown on the right. Our system performs both of these document-level tasks with a single end-to-end model.

4.3.1 Task Definition: Template Filling

Assume we are given a set of m event types (T_1, ..., T_m). Each event template contains a set of k roles (r_1, ..., r_k). For a document consisting n words x_1, x_2, ..., x_n, the system is required to extract d templates, where $d \geq 0$ (d is not given as input). Each template consists of $k + 1$ slots: the first slot represents the event type (one of T_1, ..., T_m). The rest of the k slots correspond to an event role (one of r_1, ..., r_k). The system is required to fill in entities for the corresponding role, which may be filled in as null.

4.3.2 Methodology

Our framework is illustrated in Figure 4.7. First we transform the template filling task into a sequence generation problem. Then, we train the base model on the source-target sequence pairs, and apply the model to generate the sequence; finally the sequence is transformed back to structured templates.



Event 1 Template	Attack
Perpetrator Indiv.	-
Perpetrator Org	**Zarate armed forces**
Physical Target	economic ministry
Weapon	dynamite sticks
Victim	-

Event 2 Template	Bombing
Perpetrator Indiv.	-
Perpetrator Org	**Zarate armed forces**
Physical Target	government house
Weapon	bomb
Victim	-

Template 1
Template 2
Attack <T1 REEs> **[SEP_T]** *Bombing* <T2 REEs> **[SEP_T]** ...
Generative Transformers
[CLS]
Attack, Bombing, Arson, Kidnapping, ...
[SEP_T]
(Document tokens): Several attacks were carried out in La Paz last night ...
[SEP]
[CLS] *Attack* <T1 REEs> **[SEP_T]** *Bombing* <T2 REEs> **[SEP_T]**
Source tokens
Target tokens

Figure 4.7: Our generative framework for end-to-end template filling.

4.3.3 Template Filling as Sequence Generation

We first transform the task's input and output data into specialized source and target sequence pair encodings. As shown in Figure 4.7 and below, the *source sequence* consists of the words of the document (x_1, x_2, ..., x_n) prepended with the general set of tokens representing all event/template types (T_1, ..., T_m); as well as a separator token denoting the boundary between event templates ([SEP_T]). We also add a classification token ([CLS]) and another separator token ([SEP]) at the beginning and end of this source sequence. [CLS] works as the start token, [SEP] denotes the boundary between REEs.

$$\text{[CLS]}\ T_1, ..., T_m\ \text{[SEP\_T]}$$
$$x_1, x_2, ..., x_n\ \text{[SEP]}$$

The *target sequence* consists of the concatenation of template extractions, separated by the separator token ([SEP_T]). For template i, the sub-sequence consists of its event type $T^{(i)}$ and its role-filler entity extractions $<$ Role-filler Entities $>^{(i)}$:

$$\begin{aligned} \text{[CLS]}\ & T^{(1)}, < \text{Role-filler Entities} >^{(1)} \\ \text{[SEP\_T]}\ & T^{(2)}, < \text{Role-filler Entities} >^{(2)} \\ & ... \\ \text{[SEP\_T]}\ & T^{(i)}, < \text{Role-filler Entities} >^{(i)} \\ & ... \end{aligned}$$

For the $<$ Role-filler Entities $>$ of template i, following Du et al. [2021a], we use the concatenation of target entity extractions for each role, separated by the separator token ([SEP]). Each entity is represented with its first mention's beginning (b) and end (e) tokens:

$$e^1_{1_b}, e^1_{1_e}, ..\ \text{[SEP]}\ e^2_{1_b}, e^2_{1_e}, ..\ \text{[SEP]}\ e^3_{1_b}, e^3_{1_e}, ..$$

4.3.4 Base Model and Decoding Constraints

Next we describe the base model as well as special decoding constraints for template filling.

BERT as Encoder and Decoder Our model extends upon the GRIT model for REE [Du et al., 2021a]. The base setup utilizes one BERT [Devlin et al., 2019] model for processing both the source and target tokens embeddings. To distinguish the encoder / decoder representations, it uses partial causal attention mask on the decoder side [Du et al., 2021a]. The joint sequence of source tokens' embeddings $(\mathbf{a}_0, \mathbf{a}_1, ..., \mathbf{a}_m)$ and target tokens' embeddings $(\mathbf{b}_0, \mathbf{b}_1, ..., \mathbf{b}_n)$ are passed through BERT to obtain their contextualized representations,

$$\begin{gathered}\hat{\mathbf{a}}_0, \hat{\mathbf{a}}_1, ..., \hat{\mathbf{a}}_{l_{src}}, \hat{\mathbf{b}}_0 ..., \hat{\mathbf{b}}_{l_{tgt}} \\ = \texttt{BERT}(\mathbf{a}_0, \mathbf{b}_1, ..., \mathbf{a}_{l_{src}}, \mathbf{b}_0, ..., \mathbf{b}_{l_{tgt}})\end{gathered}$$

Pointer Decoding For the final decoder layer, we replace word prediction with a simple pointer selection mechanism. For target time step t, we first calculate the dot-product between $\hat{\mathbf{b}}_t$ and $\hat{\mathbf{a}}_0, \hat{\mathbf{a}}_1, ..., \hat{\mathbf{a}}_m$,

$$c_0, c_1, ..., c_{l_{src}} = \hat{\mathbf{b}}_t \cdot \hat{\mathbf{a}}_0, \hat{\mathbf{b}}_t \cdot \hat{\mathbf{a}}_1, ..., \hat{\mathbf{b}}_t \cdot \hat{\mathbf{a}}_{l_{src}}$$

Then we apply softmax to $c_0, c_1, ..., c_{l_{src}}$ to obtain the probabilities of pointing to each source token (which may be a word or an event type), test prediction is done with greedy decoding. At each time step, argmax is applied to find the source token which has the highest probability. The decoding stops when a stop token is predicted.

$$p_0, p_1, ..., p_{l_{src}} = \text{softmax}(c_0, c_1, ..., c_{l_{src}})$$

We also add several special decoding constraints for template filling: (1) down-weighting factor (0.01) to the probability of generating [SEP] and [SEP_T], in order to calibrate recall; (2) decoding cutoff stop when it ends the k^{th} template (k =maximum number of events in one document); (3) a constraint to ensure that the pointers for the start and end token for one entity are in order.

4.3.5 Experiments

We conduct evaluations on the MUC-4 dataset [MUC-4, 1992]. MUC-4 consists of 1,700 documents with associated templates. We follow prior work in split: 1,300 documents for training, 200 documents (`TST1+TST2`) as the development set and 200 documents (`TST3+TST4`) as the test set. We use the metric for template filling [Chinchor, 1992] and, as in previous work, map predicted templates to gold templates during evaluation so as to optimize scores. We follow content-based mapping restrictions, i.e., the event type of the template is considered essential for the mapping to occur.[5] Missing template's slots are scored as missing, spurious template's slots are scored as spurious. Note that in our work, since we do not extract the set fillers other than the event/template type, they do not affect the performance.

[5] The content-based mapping restrictions were added to MUC-4 to prevent fortuitous mappings which occurred in MUC-3 [Chinchor, 1992].

Models	Event Type	PerpInd	PerpOrg	Target	Victim	Weapon
Grit-pipeline	62.28	38.40	35.36	36.30	**54.97**	53.45
DyGIE++ [Wadden et al., 2019]	61.95	32.44	25.73	**45.04**	49.48	51.60
seqTagging [Du and Cardie, 2020]	60.22	30.59	26.79	36.60	43.62	51.70
Gtt	**67.44**	**44.04**	**41.79**	32.39	54.12	**59.71**

Table 4.8: Per-slot F1 score.

Baselines and Additional Related Work As an ablation baseline, we employ a pipeline, Grit-pipeline, that first uses the Grit model for role-filler entity extraction, and then assigns event types to each of the entities as a multi-label classification problem. We assign types by transforming the problem to multi-class classification (MCC) [Spolaor et al., 2013]. As there are 6 event types (i.e., *kidnapping*, *attack*, *bombing*, *robbery*, *arson*, *forced work stoppage*) in MUC-4, we use 2^6 labels for the MCC problem.

We also compare to end-to-end baselines without modeling between-event dependencies, DyGIE++ [Wadden et al., 2019][6] is a span-enumeration based extractive model for information extraction. The model enumerates all the possible spans in the document and passes each representation through a classifier layer to predict whether the span represents certain role-filler entity and what the role is. seqTagging is a BERT-based sequence tagging model for extracting the role-fillers entities. A role-filler entity can appear in templates of different event types (e.g., "Zarate armed force" appear in both attack and bombing event). For both baselines, the prediction goal is multi-class classification. More specially, we *adapt* the DyGIE++ output layer implementation to first predict the role-filler entity's role class, and then predicts its event classes conditioned on the entity's role.

Note that Chambers [2013] and Cheung et al. [2013] propose to do event schema induction with unsupervised learning. Given their unsupervised nature, empiri-

[6] Our own re-implementation.

cally the performance is worse than supervised models [Patwardhan and Riloff, 2009]. Thus we do not add these as comparisons.

Models	P	R	F1
Grit-pipeline	**63.88**	37.56	47.31
DyGIE++ [Wadden et al., 2019]	61.90	36.33	45.79
seqTagging [Du and Cardie, 2020]	46.80	38.30	42.13
Gtt	61.69	**42.36**	**50.23***

Table 4.9: Micro-average results on the full test set.

4.3.6 Results and Analysis

Results on the full test set are shown in Table 4.9. We report the micro-average performance (precision, recall and F1). We see that our framework substantially outperforms the baseline extraction models in precision, recall and F1, with approximately a 4% F1 increase over the end-to-end baselines. It outperforms the Grit-pipeline system by around 3% F1 (* denotes $p < 0.05$).

Per-slot F1 score is reported in Table 4.8. The results demonstrate that our framework more often predicts the correct event type, performs better on PerpInd and PerpOrg, and achieves slightly worse performance with Grit-pipeline on roles that appear later in the template (i.e., Target and Victim). We also found that DyGIE++ performs better on Target, mainly due to its high precision in role assignment for spans.

Between-Event Dependencies We also show results (Table 4.10) on the subset of documents that contains more than one gold event. We see the F1 score for

Models	P	R	F1	Δ
GRIT-PIPELINE	65.17	26.05	37.22	-21.33%
DYGIE++	69.90	27.05	39.01	-14.81%
SEQTAGGING	51.00	29.06	37.02	-12.13%
GTT	56.76	38.08	45.58	-9.26%

Table 4.10: Performance on the subset of documents which contain more than one gold event. Δ: relative change of F1, as compared to the Full Test setting.

all systems drops substantially, proving the difficulty of the task, as compared to the single/no event case. When compared to the Full Test setting in Table 4.9, the baselines all increase in precision and drop substantially in recall, while our approach's precision and recall drop a little. This change is understandable, as the baseline systems are more conservative and tend to predict fewer templates. As the number of gold templates increases, the fewer templates predictions have a better chance of getting matched, but their recall drops as well.

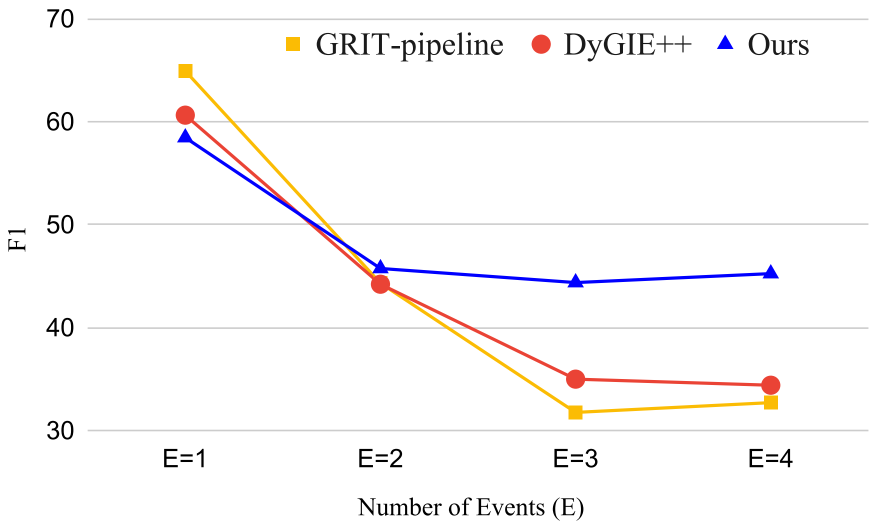


Figure 4.8: F1 on subset of documents with E events.

How performance changes when E increases In Figure 4.8, we see that when the number of gold events in the document is smaller ($E = 1, 2$), our approach performs on par with the pipeline-based and DYGIE++ baselines. However, as

E grows larger, the baselines' F1 drop significantly (e.g., over -10% as E grows from 2 to 3).

Qualitative Case Analysis Consider the input document (doc id `TST3-MUC4-0080`) below, which contains an attack and a bombing template. In the gold annotations, "Farabundo Marti National Liberation Front" acts as PerpOrg in both events. Our model correctly extracts the two events and the PerpOrg in each while DyGIE++ only predicts the attack event with its PerpOrg role entity correctly. Although Grit-pipeline gets both events correct, it failed to extract this PerpOrg entity for the second event.

> Official sources today reported that at least eight people, including soldiers, rebels, and civilians, were killed during clashes between the army and guerrillas over the past weekend in various points of the country.
>
> Military spokesmen for the 6th infantry brigade, headquartered in the eastern usulutan department, told acanefe that two rebels were killed and one wounded during a clash with government troops in San Agustin.
>
> Meanwhile, the armed forces press committee (Coprefa) reported that the bodies of two guerrillas, who were presumably killed during clashes with the army, were found by soldiers in the outskirts of Santa Tecla, in the central la libertad department.
>
> Coprefa reported that two soldiers were killed during a clash with members of the **Farabundo Marti National Liberation Front** (FMLN) in Comasagua, about 28 km to the southwest of (San) Salvador, where a rebel attack on a coffee processing plant was successfully repelled.
>
> It reported that a civilian was killed in the crossfire and that a soldier was also killed during clashes in Zaragoza, south of San Salvador, where two guerrillas were wounded.
>
> ...

> Salvadoran (red) cross sources today reported that a 48-year-old woman identified as Maria Luz Lopez was wounded last night when a powerful bomb, which damaged several businesses in (San) Salvador, exploded.
>
> The bomb was planted in a heavily commercial area of downtown (San) Salvador causing heavy property loses, according to the owners who provided no specific figures.
>
> This is the fourth dynamite attack on businesses in (San) Salvador so far in 1990.

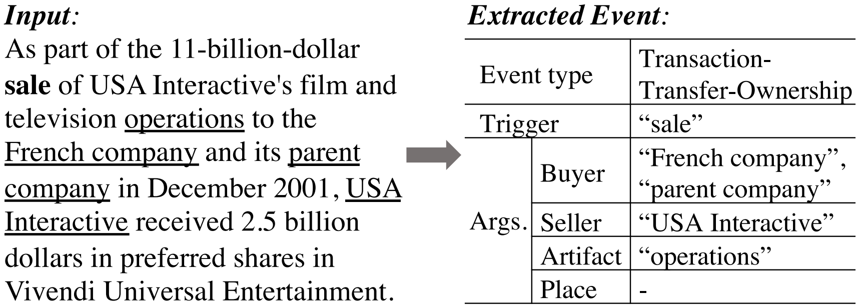
4.4 Chapter Summary

We revisit the classic NLP problem of template filling and its sub-task REE. We demonstrate that they are still challenging but more realistic problems in IE, and prior methods are not able to handle well the complex output structures and dependencies. We introduce an effective end-to-end transformer based generative model GRIT for REE, which learns the document representation and encodes the dependency between role-filler entities and between event roles. GRIT outperforms the baselines on the task and better captures the coreference linguistic phenomena; We also extend GRIT and propose the generative learning model called GTT for the full template filling task. GTT better dependencies across the document and performs substantially better on multi-event documents.

CHAPTER 5
EXTRACTION BY ANSWERING (ALMOST) NATURAL QUESTIONS

In chapter 3 and chapter 4, we've introduced modeling techniques for tackling the challenges introduced by the document-level context and complex structure. Another major challenge for IE as we mentioned in chapter 1 is the high cost of annotations, which motivates design of models with better generalizability (e.g., achieving decent performance with few annotated examples.) Next, in this chapter, we introduce a question answering based framework for information extraction. More specifically, we firstly generate template-based semantically meaningful questions and then answer questions to detect event triggers and extract corresponding arguments.

A recap for the event extraction task (ACE style) is illustrated via an example in Figure 5.1, which depicts an ownership transfer event (the *event type*), triggered by the word "sale" (the event *trigger*) and accompanied by its extracted *arguments* — text spans denoting entities that fill a set of (semantic) *roles* associated with the event type (e.g., BUYER, SELLER and ARTIFACT for ownership transfer events).



Input:
As part of the 11-billion-dollar **sale** of USA Interactive's film and television operations to the French company and its parent company in December 2001, USA Interactive received 2.5 billion dollars in preferred shares in Vivendi Universal Entertainment.

Extracted Event:

Event type		Transaction-Transfer-Ownership
Trigger		"sale"
Args.	Buyer	"French company", "parent company"
	Seller	"USA Interactive"
	Artifact	"operations"
	Place	-

Figure 5.1: Event extraction example from the ACE 2005 corpus.

Recent successful approaches to event extraction have benefited from dense features extracted by neural models [Chen et al., 2015; Nguyen et al., 2016; Liu et al., 2018] as well as contextualized lexical representations from pretrained language models [Zhang et al., 2019b; Wadden et al., 2019]. These approaches, however, exhibit two key weaknesses. First, they rely heavily on entity information for argument extraction. In particular, event argument extraction generally consists of two steps – first identifying entities and their general semantic class with trained models [Wadden et al., 2019] or a parser [Sha et al., 2018], then assigning argument roles (or no role) to each entity. Although joint models [Yang and Mitchell, 2016; Nguyen and Nguyen, 2019; Zhang et al., 2019a; Lin et al., 2020] have been proposed to mitigate this issue, error propagation [Li et al., 2013] still occurs during event argument extraction. A second weakness of neural approaches to event extraction is their inability to exploit the similarities of related argument roles across event types. For example, the ACE 2005 [Doddington et al., 2004] Conflict.Attack events and Justice.Execute events have target and person argument roles, respectively. Both roles, however, refer to a *human being who is affected* by an action. Ignoring the similarity can hurt performance, especially for argument roles with few/no examples at training time. Plus, the traditional models are unable to handle unanticipated roles at deployment time.

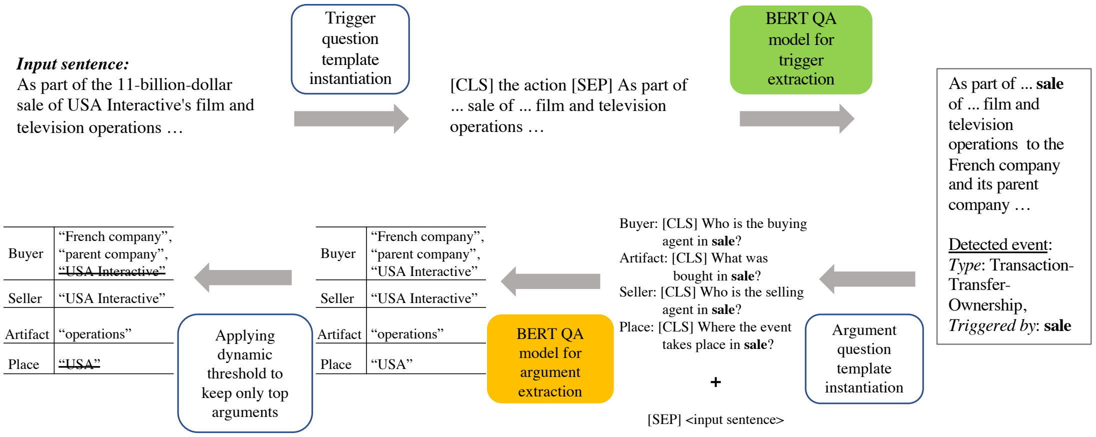
In this paper, we propose a new paradigm for the event extraction task – formulating it as a question answering (QA)/machine reading comprehension (MRC) task (**Contribution 1**). The general framework is illustrated in Figure 5.2. Using BERT [Devlin et al., 2019] as the base model for obtaining contextualized representations from the input sequences, we develop two BERT-based QA models – one for event trigger detection and the other for argument extraction. For each, we design one or more Question Templates that map the input sentence into the standard BERT input format. Thus, trigger detection becomes a request to identify "the action" or the "verb" in the input sentence and determine its event type; and argument extraction becomes a sequence of requests to identify the event's arguments, each of which is a text span in the input sentence. Details will be explained in Section 5.1. To the best of our knowledge, this is the first attempt to cast event extraction as a QA task.

Treating event extraction as QA overcomes the weaknesses in existing methods identified above (**Contribution 2**): (1) Our approach requires *no entity annotation* (gold or predicted entity information) and no entity recognition pre-step; event argument extraction is performed as an end-to-end task; (2) The question answering paradigm naturally permits the transfer of argument extraction knowledge across semantically related argument roles. We propose rule-based question generation strategies (including incorporating descriptions in annotation guidelines) for templates creation, and conduct extensive experiments to evaluate our framework on the Automatic Content Extraction (ACE) event extraction task and show empirically that the performance on both trigger and argument extraction outperform prior methods (Section 5.2.2). Finally, we show that our framework extends to the zero-shot setting – it is able to extract event arguments for unseen roles (**Contribution 3**).

5.1 Methodology

In this section, we first provide an overview for the framework (Figure 5.2), then go deeper into details of its components: question generation strategies for template creation, as well as training and inference of QA models.

5.1.1 Framework Overview



Input sentence:
As part of the 11-billion-dollar sale of USA Interactive's film and television operations …
Trigger question template instantiation
[CLS] the action [SEP] As part of … sale of … film and television operations …
BERT QA model for trigger extraction
As part of … **sale** of … film and television operations to the French company and its parent company …
Detected event:
Type: Transaction-Transfer-Ownership,
Triggered by: **sale**
Argument question template instantiation
Buyer: [CLS] Who is the buying agent in **sale**?
Artifact: [CLS] What was bought in **sale**?
Seller: [CLS] Who is the selling agent in **sale**?
Place: [CLS] Where the event takes place in **sale**?
+
[SEP] <input sentence>
BERT QA model for argument extraction
Applying dynamic threshold to keep only top arguments

Buyer	"French company", "parent company", "USA Interactive"
Seller	"USA Interactive"
Artifact	"operations"
Place	"USA"

Buyer	"French company", "parent company", ~~"USA Interactive"~~
Seller	"USA Interactive"
Artifact	"operations"
Place	~~"USA"~~

Figure 5.2: Our framework for event extraction (ACE style).

Our QA framework for event extraction relies on two sets of Question Templates that map an input sentence to a suitable input *sequence* for two instances of a standard pre-trained bidirectional transformer (BERT [Devlin et al., 2019]). The first of these, BERT_QA_Trigger (green box in Figure 5.2), extracts from the input sentence the event trigger which is a single token, and its type (one of a fixed set of pre-defined event types). The second QA model, BERT_QA_Arg (orange box in Figure 5.2), is applied to the input sequence, the extracted event trigger and its event type to iteratively identify candidate event arguments (spans of text) in the input sentence. Finally, a dynamic threshold is applied to the extracted candidate arguments, and only the arguments with probability above the threshold are retained.

The input sequences for the two QA models share a standard BERT-style format:

[CLS] <question> [SEP] <sentence> [SEP]

where [CLS] is BERT's special classification token, [SEP] is the special token to denote separation, and ¡sentence¿ is the tokenized input sentence. We provide details on how to obtain the <**question**> in Section 5.1.2. Details on the QA models and the inference process will be explained in Section 5.1.3.

5.1.2 Question Generation Strategies

For our QA-based framework for event extraction to be easily moved from one domain to the other, we concentrated on developing question generation strategies that not only worked well for the task, but can be quickly and easily implemented. For event trigger detection, we experiment with a set of four fixed templates – "what is the trigger", "trigger", "action", "verb". Basically, we use the fixed literal phrase as the question. For example, if we choose the "action" template, the input sequence for the example sentence in Figures 5.1 and 5.2 is instantiated as:

[CLS] action [SEP] As part of the 11-billion-dollar sale ... [SEP]

As for event argument extraction, we design three templates with argument role name, basic argument based question and annotation guideline based question, respectively:

Argument	Template 1 (Role name)	Template 2 (Type + Role question)	Template 3 (Annotation guideline question)
Artifact	artifact	What is the artifact?	What is being transported?
Agent	agent	Who is the agent?	Who is responsible for the transport event?
Vehicle	vehicle	What is the vehicle?	What is the vehicle used?
Origin	origin	What is the origination?	Where the transporting originated?
Destination	destination	What is the destination?	Where the transporting is directed?

Table 5.1: Arguments (of event type MOVEMENT.TRANSPORT) and corresponding questions from three templates. "in <trigger>" is not added to the questions in this example.

- **Template 1 (Role Name)** For this template, ¡question¿ is simply instantiated with the argument role name (e.g., artifact, agent, place).

- **Template 2 (Type + Role)** Instead of directly using the argument role name (<role name>) as the question, we first determine the argument role's general semantic type — one of person, place, other; and construct the associated "WH" word question – *who* for person, *where* for place and *what* for all other cases, of the following form:

  <WH_word> is the <role name> ?

  Examples are shown in Table 1 for the arguments of event type Movement.Transport. By adding the WH word, more semantic information is included as compared to Template 1.

- **Template 3 (Incorporating Annotation Guidelines)** To incorporate even more semantic information and make the question more natural sounding, we utilize the descriptions of each argument role provided in the ACE annotation guidelines for events [Linguistic Data Consortium, 2005] for generating the questions.

- **+ "in <trigger>"** Finally, for each template type, it is possible to encode the trigger information by adding "in <trigger>" at the end of the question (where <trigger> is instantiated with the real trigger token obtained from the trigger detection phase). For example, the Template 2 question incorporating trigger information would be:

  <WH_word> is the <argument> in <trigger>?

To help better understand all the strategies above, Table 5.1 presents an example for argument roles of event type Movement.Transport. We see that the annotation guideline based questions are more natural and encode more semantics about a given argument role, than the simple Type + Role question "what is the artifact?".

5.1.3 Question Answering Models

We use BERT [Devlin et al., 2019] as the base model for getting contextualized representations for the input sequences for both BERT_QA_Trigger and BERT_QA_Arg. After the instantiation with question templates the sequences are of format [CLS] <question> [SEP] <sentence> [SEP].

Then we get the contextualized representations of each token for trigger detection and argument extraction with $\texttt{BERT}_{Tr}$ and $\texttt{BERT}_{Arg}$, respectively. For the input sequence $(e_1, e_2, ..., e_N)$ prepared for trigger detection, we have:

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_N]$$

$$\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_N = \texttt{BERT}_{Tr}(e_1, e_2, ..., e_N)$$

For the input sequence $(a_1, a_2, ..., a_M)$ prepared for argument span extraction, we have:

$$\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_M]$$

$$\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_M = \texttt{BERT}_{Arg}(a_1, a_2, ..., a_M)$$

The output layer of each QA model, however, differs: BERT_QA_Trigger predicts the event type for each token in sentence (or None if it is not an event trigger), while BERT_QA_Arg predicts the start and end offsets for the argument span with a different decoding strategy.

More specifically, for **trigger prediction**, we introduce a new parameter matrix $\mathbf{W}_{tr} \in \mathbb{R}^{H \times T}$, where H is the hidden size of the transformer and T is the number of event types plus one (for non-trigger tokens). `softmax` normalization is applied across the T types to produce P_{tr}, the probability distribution across the event types:

$$P_{tr} = \texttt{softmax}(\mathbf{E}\mathbf{W}_{tr}) \in \mathbb{R}^{T} \times N$$

At test time, for trigger detection, to obtain the type for each token $e_1, e_2, ..., e_N$, we simply apply argmax to P_{tr}.

For **argument span prediction**, we introduce two new parameter matrices $\mathbf{W}_s \in \mathbb{R}^{H \times 1}$ and $\mathbf{W}_e \in \mathbb{R}^{H \times 1}$; softmax normalization is then applied across the input tokens $a_1, a_2, ..., a_M$ to produce the probability of each token being selected as the start/end of the argument span:

$$P_s(i) = \texttt{softmax}(\mathbf{a}_i \mathbf{W}_s)$$
$$P_e(i) = \texttt{softmax}(\mathbf{a}_i \mathbf{W}_e)$$

To train the models (BERT_QA_Trigger and BERT_QA_Arg), we minimize the negative log-likelihood loss for both models, parameters are updated during the training process. In particular, the loss for the argument extraction model is the sum of two parts: the start token loss and end end token loss. For the training examples with no argument span (no answer case), we minimize the start and end probability of the first token of the sequence ([CLS]).

$$\mathcal{L}_{arg} = \mathcal{L}_{arg\_start} + \mathcal{L}_{arg\_end}$$

Inference with Dynamic Threshold for Argument Spans At test time, predicting the argument spans is more complex – for each argument role, there can be *several* or *no* spans to be extracted. After the output layer, we have the probability of each token $a_i \in (a_1, a_2, ..., a_M)$ being the start ($P_s(i)$) and end ($P_e(i)$) of the argument span.

Algorithm 1: Harvesting Argument Spans Candidates

```
Input  : $P_s(i)$, where $i \in \{1, ..., M\}$,
         $P_e(i)$, where $i \in \{1, ..., M\}$
Output: valid candidate spans for the argument role
for $start \leftarrow 1$ to $M$ do
    for $end \leftarrow 1$ to $M$ do
        if $start$ or $end$ not in the input sentence then continue;
        if $end - start + 1 > MaxSpanLength$ then continue;
        if $P_s(start) < P_s([CLS])$ or $P_e(end) < P_e([CLS])$ then continue;
        // add the valid candidate span to the set
        $score \leftarrow P_s(start) + P_e(end)$;
        $no\_ans\_score \leftarrow P_s(1) + P_e(1) - score$;
        $candidates$.add($[start, end, no\_ans\_score]$)
    end
end
```

Algorithm 2: Automatic Filtering on Argument Candidates

```
Input  : $dev\_candidates(i)$, $i \in \{1, ..., dev\_n\}$,
         $test\_candidates(i)$, $i \in \{1, ..., test\_n\}$.
Output: A set of top arguments from test_candidates
// get the best dynamic threshold
sort($dev\_candidates, key = no\_ans\_score$);
$best\_thresh \longleftarrow 0$;
$best\_res \longleftarrow 0$;
for $i \leftarrow 1$ to $dev\_n$ do
    $thresh \leftarrow dev\_candidates(i).no\_ans\_score$;
    $result \leftarrow$ eval($dev\_candidates$ with $no\_ans\_score <= thresh$);
    if $result > best\_res$ then $best\_thresh \leftarrow thresh$;
      $best\_res \leftarrow result$;
end
// apply the best threshold
$final\_arguments \longleftarrow \{\}$;
for $i \leftarrow 1$ to $test\_n$ do
    if $test\_candidates(i).no\_ans\_score <= best\_thresh$ then
      $final\_arguments$.add($test\_candidates(i)$);
end
```

Firstly, we run an algorithm to harvest all valid argument spans candidates for each argument role (Algorithm 1). Basically, we:

1. Enumerate all the possible combinations of start offset (*start*) and end offset (*end*) of the argument spans (line 1–2);
2. Eliminate the spans not satisfying the constraints: start and end token must be within the sentence; the length of the span should be shorter than a maximum length constraint; Argument spans should have larger probability than the probability of "no argument" (which is stored at the [CLS] token) (line 3–5);
3. Calculate the relative no answer score (*no_ans_score*) for the candidate span and add the candidate to list (line 6–8).

Then we run another algorithm to filter out candidate arguments that should not be included (Algorithm 2). More specifically, we obtain a probability threshold (*best_thresh*) that helps achieve best evaluation results on the dev set (line 1–9) and keep only those arguments with *no_ans_score* smaller than the threshold (line 10–13). With the dynamic threshold for determining the number of arguments to be extracted for each role[1], we avoid adding a (hard) hyperparameter for this purpose.

Another easier way to get final argument predictions is to directly include all the candidates with $no\_ans\_score < 0$, which does not require tuning the dynamic threshold *best_thresh*.

[1] Each role has a separate threshold.

5.2 Experiments

5.2.1 Dataset and Evaluation Metric

We conduct experiments on the ACE 2005 corpus [Doddington et al., 2004], it contains documents crawled between year 2003 and 2005 from a variety of areas such as newswire (nw), weblogs (wl), broadcast conversations (bc) and broadcast news (bn). The part that we use for evaluation is fully annotated with 5,272 event triggers and 9,612 arguments. We use the same data split and pre-processing step as in the prior works [Zhang et al., 2019b; Wadden et al., 2019].

As for evaluation, we adopt the same criteria defined in Li et al. [2013]: An event trigger is correctly identified (ID) if its offsets match those of a gold-standard trigger; and it is correctly classified if its event type (33 in total) also matches the type of the gold-standard trigger. An event argument is correctly identified (ID) if its offsets and event type match those of any of the reference argument mentions in the document; and it is correctly classified if its semantic role (22 in total) is also correct. Though our framework does not involve the trigger/argument identification step and tackles the identification + classification in an end-to-end way, we still report the trigger/argument identification's results to compare to prior work. It could be seen as a more lenient evaluation metric, as compared to the final trigger detection and argument extraction metric (ID + Classification), which requires both the offsets and the type to be correct. All the aforementioned elements are evaluated using precision (denoted as P), recall (denoted as R) and F1 scores (denoted as F1).

5.2.2 Results

Evaluation on ACE Event Extraction We compare our framework's performance to a number of prior competitive models: **dbRNN** [Sha et al., 2018] is an LSTM-based framework that leverages the dependency graph information to extract event triggers and argument roles. **Joint3EE** [Nguyen and Nguyen, 2019] is a multi-task model that performs entity recognition, trigger detection and argument role assignment by shared BiGRU hidden representations. **GAIL** [Zhang et al., 2019b] is an ELMo-based model that utilizes a generative adversarial network to help the model focus on harder-to-detect events. **DYGIE++** [Wadden et al., 2019] is a BERT-based framework that models text spans and captures within-sentence and cross-sentence context. **OneIE** [Lin et al., 2020] is a joint neural model for extraction with global features.[2]

	Trigger Identification			Trigger ID + Classification		
	P	R	F1	P	R	F1
dbRNN [Sha et al., 2018]	-	-	-	74.10	69.80	71.90
Joint3EE [Nguyen and Nguyen, 2019]	70.50	74.50	72.50	68.00	71.80	69.80
GAIL-ELMo [Zhang et al., 2019b]	76.80	71.20	73.90	74.80	69.40	72.00
DYGIE++, BERT + LSTM [Wadden et al., 2019]	-	-	-	-	-	68.90
DYGIE++, BERT FineTune [Wadden et al., 2019]	-	-	-	-	-	69.70
Our BERT FineTune	69.77	76.18	72.84	67.15	73.20	70.04
BERT_QA_Trigger (best trigger question strategy)	74.29	77.42	75.82	71.12	73.70	**72.39**

Table 5.2: Trigger detection results.

In Table 5.2, we present the comparison of models' performance on trigger detection. We also implement a BERT fine-tuning baseline and it reaches nearly same performance as its counterpart in DYGIE++. We observe that our BERT_QA_Trigger model with the best trigger questioning strategy reaches comparable (better) performance with the baseline models.[3]

[2] Slightly different from our and Wadden et al. [2019]'s data pre-processing, OneIE skips lines before the <text> tag (e.g., headline, datetime).

[3] Note that OneIE is concurrent to our work and reports better performance. On trigger detection, it reaches 74.7 F1 as compare to our 72.39. On argument extraction (affected by trigger detection), it reaches 56.8 as compared to our 53.31.

	Argument Identification			Argument ID + Classification		
	P	R	F1	P	R	F1
dbRNN [Sha et al., 2018]	-	-	57.20	-	-	50.10
Joint3EE [Nguyen and Nguyen, 2019]	-	-	-	52.10	52.10	52.10
GAIL-ELMo [Zhang et al., 2019b]	63.30	48.70	55.10	61.60	45.70	52.40
DYGIE++, BERT + LSTM [Wadden et al., 2019]	-	-	54.10	-	-	51.40
DYGIE++, BERT + LSTM ensemble [Wadden et al., 2019]	-	-	55.40	-	-	52.50
BERT_QA_Arg (annot. guideline question template)	58.02	50.69	54.11	56.87	49.83	53.12*
w/o dynamic threshold	53.39	54.69	54.03	50.81	52.78	51.77
BERT_QA_Arg (ensemble argument question template 2&3)	58.90	52.08	55.29	56.77	50.24	**53.31**

Table 5.3: Argument extraction results. * indicates statistical significance ($p < 0.05$).

In Table 5.3, we present the comparison between our model and baseline systems on argument extraction. Notice that the performance of argument extraction is directly affected by trigger detection. Because argument extraction correctness requires the trigger to which the argument refers to be correctly identified and classified. We can observe: (1) Our BERT_QA_Arg model with the best argument question generation strategy (annotation guideline based questions) outperforms prior work significantly, although it uses no entity recognition resources; (2) Drop of F1 performance from argument identification (correct offset) to argument ID + classification (both correct offset and argument role) is only around 1%, while the gap is around 3% for prior models which rely on entity recognition and a multi-step process for argument extraction. This once again demonstrates the benefit of our new formulation for the task as question answering.

To better understand how the dynamic threshold is affecting our framework's performance. We conduct an ablation study on this (Table 5.3) and find that the threshold increases the precision and the general F1 substantially. The last row in the table shows the test time ensemble performance of the predictions from BERT_QA_Arg trained with template 2 question, and another BERT_QA_Arg trained with template 3 question (the two relatively better questioning strategies). The ensemble system outperforms the non-ensemble system in both precision and recall, demonstrating benefits from both templates.

Evaluation on Unseen Argument Roles To verify how our formulation provides advantages for extracting arguments with unseen argument roles (similar to the zero-shot relation extraction setting in Levy et al. [2017]), we conduct another experiment, where we keep 80% of the argument roles (16 roles) seen at training time, and 20% (6 roles) only seen at test time. Specifically, the unseen roles are "Vehicle, Artifact, Target, Victim, Recipient, Buyer". Notice that during training, we use the subset of sentences from the training set, which are known to contain arguments of seen roles as positive examples. At test time, we evaluate the models on the subset of sentences from the test set, which contains arguments of unseen roles.[4]

	Argument ID + Classification		
	P	R	F1
Random NE	26.61	24.77	25.66
GAIL [Zhang et al., 2019b]	-	-	-
Our model			
w/ Role name	73.83	53.21	61.85
w/ Type + Role Q	77.18	55.05	64.26
w/ Annot. Guideline Q	78.52	59.63	**67.79**

Table 5.4: Evaluation on sentences containing unseen argument roles.

Table 5.4 presents the results. **Random NE** is our random baseline that selects a named entity in the sentence, it has a reasonable performance of near 25%. Prior models such as **GAIL** are not capable of handling the unseen roles. **ZSTE** [Huang et al., 2018] is a framework for zero-shot transfer learning of event extraction with AMR. It maps each parsed candidate span to a specific type in a target event ontology. Its argument extraction results are affected by AMR performance and

[4] We omit the trigger detection phase in this evaluation.

their reported F1 is around 20-30% in their evaluation setting.

Using our QA-based framework, as we leverage more semantic information and naturalness into the question (from question template 1 to 2, to 3), both the precision and recall increase substantially.

5.3 Further Analysis

5.3.1 Influence of Question Templates

To investigate how the question generation strategies affect the performance of event extraction, we perform experiments on trigger and argument extractions with different strategies, respectively.

	Trigger ID + Classification		
	P	R	F1
leaving empty	67.15	73.20	70.04
“what is the trigger”	70.15	69.98	70.06
“what happened”	70.53	69.48	70.00
“trigger”	69.73	71.46	70.59
“action”	72.25	71.71	71.98
“verb”	71.12	73.70	**72.39**

Table 5.5: Effect of questioning on trigger detection.

In Table 5.5, we try different fixed questions for trigger detection. By “leaving empty”, we mean instantiating the **question** with empty string.[5] There’s no substantial gap between different alternatives. By using “verb” as the question, our BERT_QA_Trigger model achieves best performance (measured by F1 score).

[5] In this case, the model degrades to a token classification model, which matches our BERT FineTune baseline’s performance.

The QA model also encodes the semantic *interactions* between the fixed question ("verb") and the sentence, this explains why BERT_QA_Trigger is better than BERT FineTune in trigger detection.

	Predicted Triggers						Gold Triggers					
	Argument Identification			Argument ID + C			Argument Identification			Argument ID + C		
Question	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Role name	47.50	51.22	49.29	44.85	48.78	46.74	56.12	67.01	61.09	51.95	63.19	57.02
+ in ¡trigger¿	53.86	51.91	52.87	51.63	50.17	50.89	69.00	64.76	66.81	64.70	61.28	62.94
Type + Role question	51.02	47.74	49.33	48.64	45.83	47.19	60.31	62.15	61.22	57.17	59.20	58.17
+ in ¡trigger¿	54.61	50.69	52.58	52.98	48.96	50.89	70.38	62.85	66.40	67.55	60.59	63.88
Annot. guideline question	51.17	51.22	51.19	48.99	49.83	49.40	60.03	68.40	63.94	57.08	65.97	61.21
+ in ¡trigger¿	58.02	50.69	54.11	56.87	49.83	**53.12**	71.17	65.45	68.19	67.88	63.02	**65.36**

Table 5.6: Influence of question generation strategies on argument extraction.

The comparison between different question generation strategies for argument extraction is even more interesting. In Table 5.6, we present the results in two settings: event argument extraction with predicted triggers (the same setting as in Table 5.3), and with gold triggers. In summary, we find that:

- *Adding "in <trigger>" after the question consistently improves the performance.* It serves as an indicator for what/where the trigger is in the input sentence. Without adding the "in <trigger>", for each template (1, 2 & 3), the F1 of models' predictions drop around 3 percent when given predicted triggers, and more when given gold triggers.

- *Our template 3 questioning strategy which is most natural achieves the best performance.* As we mentioned earlier, template 3 questions are based on descriptions for argument roles in the annotation guideline, thus encoding more semantic information about the role name. And this corresponds to the accuracy of models' predictions – template 3 is more effective than templates 1&2 in both with "in <trigger>" and without "in <trigger>" settings. What's more, we observe that template 2 (adding a WH_word to form the questions) achieves better performance than the template 1 (directly using argument role name).

5.3.2 Error Analysis

We further conduct error analysis and provide a number of representative examples. Table 5.7 summarizes error statistics for trigger detection and argument extraction.

Missing	Spurious	Wrong Type
46.08%	45.62%	8.29%

same number		more	less
exact match	not exact match		
14.48%	17.21%	13.93%	54.37%

Table 5.7: Trigger errors (upper table) and argument errors (lower table).

For event triggers, the majority of the errors relate to missing or spurious predictions, and only 8.29% involve misclassified event types (e.g., an Elect event is mistaken for a Start-Position event). For event arguments, on the sentences that come with at least one event in gold data, our framework extracts more arguments only around 14% of the cases. Most of the time (54.37%), our framework extracts fewer arguments than it should; this corresponds to the results in Table 5.3, where the precision of our models are higher. In around 30% of the cases, our framework extracts the same number of arguments as in the gold data, almost half of which match exactly the gold arguments.

After examining the example predictions, we find that reasons for errors can be mainly divided into the following categories:

- *More complex sentence structures.* In the following example, the input sentence has multiple clauses, each with trigger and arguments (such as when triggers are partial or elided). Our model is capable of also extracting "Tom" as another ENTITY of the CONTACT.MEET event in the first example:

  > [She]$_{\text{ENTITY}}$ **visited** the store and [Tom]$_{\text{ENTITY}}$ did too.

  But in the second example, when there is a higher-order event expressed spanning events in nested clauses, our model did not extract the entire VICTIM correctly, which shows the difficulty of handling complex clause structures.

  > Canadian authorities arrested two Vancouver-area men on Friday and charged them in the **deaths** of [329 passengers and crew members of an Air-India Boeing 747 that blew up over the Irish Sea in 1985, en route from Canada to London]$_{\text{VICTIM}}$.

- *Lack of reasoning with document-level context.* In the sentence "MCI must now seize additional assets owned by Ebbers, to secure the **loan**." There is a TRANSFER-MONEY event triggered by loan, with MCI as the GIVER and Ebbers, the RECIPIENT. In the previous paragraph, it's mentioned that "Ebbers failed to make repayment of certain amount of money on the loan from MCI." Without this context, it is hard to determine that Ebbers should be the recipient of the loan.

- *Lack of knowledge to obtain exact boundary of the argument span.* For example, in "Negotiations between Washington and Pyongyang on their nuclear dispute have been set for April 23 in Beijing ...", for the ENTITY role, two argument spans should be extracted ("Washington" and "Pyongyang"). While our framework predicts the entire "Washington and Pyongyang" as the argument span. Although there's an overlap between the prediction and gold-data, the model gets no credit for it.

- *Data and lexical sparsity.* In the following two examples, our model fails to detect the triggers of type END-POSITION. "Minister Tony Blair said **ousting** Saddam Hussein now was key to solving similar crises." "There's no indication if Erdogan would **purge** officials who opposed letting in the troops." It's partially due to they not being seen during training as triggers. "ousting" is a rare word and is not in the tokenizers' vocabulary. Purely inferring from the sentence context is hard to make the correct prediction.

5.4 Related Work

Event Extraction Most event extraction research has focused on the 2005 Automatic Content Extraction (ACE) sentence-level event task [Walker et al., 2006]. In recent years, continuous representations from convolutional neural networks [Nguyen and Grishman, 2015; Chen et al., 2015] and recurrent neural networks [Nguyen et al., 2016] have been proved to help substantially for pipeline-based classifiers by automatically extracting features. To mitigate the effect of error propagation, joint models have been proposed for event extraction. Yang and Mitchell [2016] consider structural dependencies between events and entities, which requires heavy feature engineering to capture discriminative information. Nguyen and Nguyen [2019] propose a multitask model that performs entity recognition, trigger detection and argument role prediction by sharing BiGRU hidden representations. Zhang et al. [2019a] utilize a neural transition-based extraction framework [Zhang and Clark, 2011], which requires specially designed transition actions. It still requires recognizing entities during decoding, though entity recognition and argument role prediction are done jointly.

These methods generally perform **trigger detection → entity recognition → argument role assignment** during decoding. Different from the works above, our framework completely bypasses the entity recognition stage (thus no annotation resources for NER needed), and directly tackles event argument extraction. Also related to our work includes DYGIE++ [Wadden et al., 2019] – it models the entity/argument spans (with start and end offset) instead of labeling with the BIO scheme. Different from our work, its learned span representations are later used to predict the entity/argument type. While our QA model directly extracts the spans for certain argument role types. Contextualized representations produced by pre-trained language models [Peters et al., 2018; Devlin et al., 2019] have been shown to be helpful for event extraction [Zhang et al., 2019b; Wadden et al., 2019] and question answering [Rajpurkar et al., 2016]. The attention mechanism helps capture relationships between tokens in the question and input sequence tokens. We use BERT in our framework for capturing these semantic relationships.

Machine Reading Comprehension (MRC) Span-based MRC tasks involve extracting a span from a paragraph [Rajpurkar et al., 2016] or multiple paragraphs [Joshi et al., 2017; Kwiatkowski et al., 2019]. Recently, there have been explorations on formulating NLP tasks as a question answering problem. McCann et al. [2018] proposes natural language decathlon challenge (decaNLP), which consists of ten tasks (e.g., machine translation, summarization, question answering). They cast all tasks as question answering over a context and propose a general model for this. In the information extraction literature, Levy et al. [2017] propose the zero-shot relation extraction task and reduce the task to answering crowdsourced reading comprehension questions. Li et al. [2019b] casts entity-relation extraction as a multi-turn question answering task. Their questions lack diversity and naturalness. For example for the PART-WHOLE relation, the template question is “find Y that belongs to X”, where X is instantiated with the pre-given entity. The follow-up work for named entity recognition from Li et al. [2019a] propose better query strategies incorporating synonyms and examples. Different from the works above, we focus on the more complex event extraction task, which involves both trigger detection and argument extraction. Our generated questions for extracting event arguments are somewhat more natural (incorporating descriptions from annotation guidelines) and leverage trigger information.

Question Generation To generate question templates 2 & 3 (Type + Role question and annotation guideline based question) which are more natural, we draw insights from the literature of automatic rule-based question generation [Heilman and Smith, 2010]. Heilman [2011] propose to use linguistically motivated rules for WH word (question phrase) selection. In their more general case of question generation from sentences, answer phrases can be noun phrases, prepositional phrases, or subordinate clauses. Complicated rules are designed with help from the superTagger [Ciaramita and Altun, 2006]. In our case, event arguments are mostly noun phrases and the rules are simpler – “who” for person, “where” for place and “what” for all other types of entities. We sample around 10 examples from the development set to determine the entity type of each argument role. In the future, it will be interesting to investigate how to utilize machine learning-based question generation methods [Du et al., 2017]. They would be more beneficial for the setting where the schema/ontology contains a large number of argument types, as well as generating synthetic QA pairs for data augmentation.

5.5 Chapter Summary

In this chapter, we introduce a new paradigm for event extraction based on question answering. We investigate how the question generation strategies affect the performance of our framework on both trigger detection and argument span extraction, and find that more natural questions lead to better performance. Our framework outperforms prior works on the ACE 2005 benchmark, and is capable of extracting event arguments of roles not seen at training time.

CHAPTER 6

GENERATING PARAGRAPH-LEVEL QA PAIRS

As we've shown in chapter 5, formulating information extraction (especially the more complex event extraction) problems as question answering is beneficial and help tackle the limited annotations challenge. Also demonstrated by Liu et al. [2020], the QA formulation for IE has even larger advantage when additional QA pairs are available to be leveraged during training. Motivated by this observation, in this chapter, we discuss further how to generate/harvest context-sensitive synthetic QA pairs with ML-based methods. Thus, the problem that we tackle in this chapter is question generation, instead of information extraction.

Other than being helpful for IE, question generation (QG) frameworks can also be utilized to provide additional data for other tasks in NLP (e.g., question answering and machine reading comprehension). Recently, there has been a resurgence of work in NLP on reading comprehension [Hermann et al., 2015; Rajpurkar et al., 2016; Joshi et al., 2017] with the goal of developing systems that can answer questions about the content of a given passage or document. Large-scale QA datasets are indispensable for training expressive statistical models for this task and play a critical role in advancing the field. And there have been a number of efforts in this direction. Miller et al. [2016], for example, develop a dataset for open-domain question answering; Rajpurkar et al. [2016] and Joshi et al. [2017] do so for reading comprehension (RC); and Hill et al. [2015] and Hermann et al. [2015], for the related task of answering cloze questions [Winograd, 1972; Levesque et al., 2011]. To create these datasets, either crowdsourcing or *(semi-)synthetic* approaches are used. The (semi-)synthetic datasets (e.g., Hermann et al. [2015]) are large in size and cheap to obtain; however, they do not share the same characteristics as explicit QA/RC questions [Rajpurkar et al., 2016]. In comparison, high-quality *crowdsourced* datasets are much smaller in size, and the annotation process is quite expensive because the labeled examples require expertise and careful design [Chen et al., 2016].

Recently there have been investigations on methods that can automatically generate high-quality question-answer pairs. Serban et al. [2016] propose the use of recurrent neural networks to generate QA pairs from structured knowledge resources such as Freebase. Their work relies on the existence of automatically acquired KBs, which are known to have errors and suffer from incompleteness. They are also non-trivial to obtain. In addition, the questions in the resulting dataset are limited to queries regarding a single fact (i.e., tuple) in the KB. Motivated by the need for large scale QA pairs and the limitations of recent work, we investigate methods that can automatically "harvest" (generate) question-answer pairs from raw text/unstructured documents, such as Wikipedia-type articles.

Other recent work along these lines [Du et al., 2017; Zhou et al., 2017] (see Section 6.1) has proposed the use of attention-based recurrent neural models trained on the crowdsourced SQuAD dataset [Rajpurkar et al., 2016] for question generation. While successful, the resulting QA pairs are based on information from a single sentence. As described in Du et al. [2017], however, nearly 30% of the questions in the human-generated questions of SQuAD rely on information beyond a single sentence. For example, in Figure 6.1, the second and third questions require coreference information (i.e., recognizing that "His" in sentence 2 and "He" in sentence 3 both corefer with "Tesla" in sentence 1) to answer them.

Thus, our research studies methods for incorporating coreference information into the training of a question generation system. In particular, we propose gated

Paragraph:
(1) *Tesla* was renowned for *his* achievements and showmanship, eventually earning *him* a reputation in popular culture as an archetypal ”mad scientist”. (2) *His* patents earned *him* a considerable amount of money, much of which was used to finance *his* own projects with varying degrees of success. (3) *He* lived most of his life in a series of New York hotels, through *his* retirement. (4) *Tesla* died on 7 January 1943. ...

Questions:

– What was Tesla’s reputation in popular culture?
mad scientist

– How did Tesla finance his work?
patents

– Where did Tesla live for much of his life?
New York hotels

Figure 6.1: Example input from the fourth paragraph of a Wikipedia article on *Nikola Tesla*, along with the natural questions and their answers from the SQuAD [Rajpurkar et al., 2016] dataset. We show in italics the set of mentions that refer to Nikola Tesla — *Tesla*, *him*, *his*, *he*, etc.

Coreference knowledge for **N**eural **Q**uestion **G**eneration (**CorefNQG**), a neural sequence model with a novel gating mechanism that leverages continuous representations of *coreference clusters* — the set of mentions used to refer to each entity — to better encode linguistic knowledge introduced by coreference, for paragraph-level question generation.

In an evaluation using the SQuAD dataset, we find that CorefNQG enables better question generation. It outperforms significantly the baseline neural sequence models that encode information from a single sentence, and a model that encodes *all* preceding context and the input sentence itself. When evaluated on only the portion of SQuAD that requires coreference resolution, the gap between our system and the baseline systems is even larger.

By applying our approach to the 10,000 top-ranking Wikipedia articles, we obtain a question answering/reading comprehension dataset with over one million synthetic QA pairs; we provide a qualitative analysis in Section 6.5.

6.1 Related Work

6.1.1 QG for IE

One important step in formulation (event) extraction tasks as QA/MRC is to turn the argument roles names into natural language questions. One mainstream is to use *human designed templates* for extractive tasks: relation extraction [Levy et al., 2017], semantic role labeling (QA-SRL) [FitzGerald et al., 2018], entity and relation extraction [Li et al., 2019b,a], as well as our work presented in chapter 5 on event extraction. However, Liu et al. [2020] argued that the template-based question generation may not be expressive enough to instruct an MRC model to find answers. They formulate it as an unsupervised translation task [Lample et al., 2018], which transforms a descriptive statement into a question-style expression with no parallel resources. When the descriptive sentences and questions are aligned, the problem can be directly formulated as a *learning-based* question generation task in a similar way, which we discuss in this chapter.

Apart from generating questions for each argument role, the generated synthetic QA pairs can also be utilized as additional training data. As shown by Liu et al. [2020], QA pairs from other domain can also help with performance of IE tasks.

6.1.2 Question Generation

Since the work by Rus et al. [2010], question generation (QG) has attracted interest from both the NLP and NLG communities. Most early work in QG employed rule-based approaches to transform input text into questions, usually requiring the application of a sequence of well-designed general rules or templates [Mitkov and Ha, 2003; Labutov et al., 2015]. Heilman and Smith [2010] introduced an overgenerate-and-rank approach: their system generates a set of questions and then ranks them to select the top candidates. Apart from generating questions from raw text, there has also been research on question generation from symbolic representations [Yao et al., 2012; Olney et al., 2012].

With the recent development of deep representation learning and large QA datasets, there has been research on recurrent neural network based approaches for question generation. Serban et al. [2016] used the encoder-decoder framework to generate QA pairs from knowledge base triples; Reddy et al. [2017] generated questions from a knowledge graph; Du et al. [2017] studied how to generate questions from sentences using an attention-based sequence-to-sequence model and investigated the effect of exploiting sentence- vs. paragraph-level information. Du and Cardie [2017] proposed a hierarchical neural sentence-level sequence tagging model for identifying question-worthy sentences in a text passage. Finally, Duan et al. [2017a] investigated how to use question generation to help improve question answering systems on the answer sentence selection subtask.

In comparison to the related methods from above that generate questions from raw text, our method is different in its ability to take into account contextual information beyond the sentence-level by introducing coreference knowledge.

6.2 Task Definition

Our goal is to harvest high quality question-answer pairs from the paragraphs of an article of interest. In our task formulation, this consists of two steps: **candidate answer extraction** and **answer-specific question generation**. Given an input paragraph, we first identify a set of *question-worthy* candidate answers $ans = (ans_1, ans_2, ..., ans_l)$, each is a span of text as denoted in color in Figure 6.1. For each candidate answer ans_i, we then aim to generate a question Q — a sequence of tokens $y_1, ..., y_N$ — based on the sentence S that contains candidate ans_i such that:

- Q asks about an aspect of ans_i that is of potential interest to a human;
- Q might rely on information from sentences that precede S in the paragraph.

Mathematically then,

$$Q = \arg\max_{Q} P\left(Q|S, C\right) \tag{6.1}$$

where $P(Q|S,C) = \prod_{n=1}^{N} P\left(y_n|y_{<n}, S, C\right)$, where C is the set of sentences that precede S in the paragraph.

6.3 Methodology

In this section, we introduce our framework for harvesting the question-answer pairs. As described above, it consists of the question generator CorefNQG (Figure 6.2) and a candidate answer extraction module. During test/generation time, we (1) run the answer extraction module on the input text to obtain answers, and then (2) run the question generation module to obtain the corresponding questions.

6.3.1 Question Generation

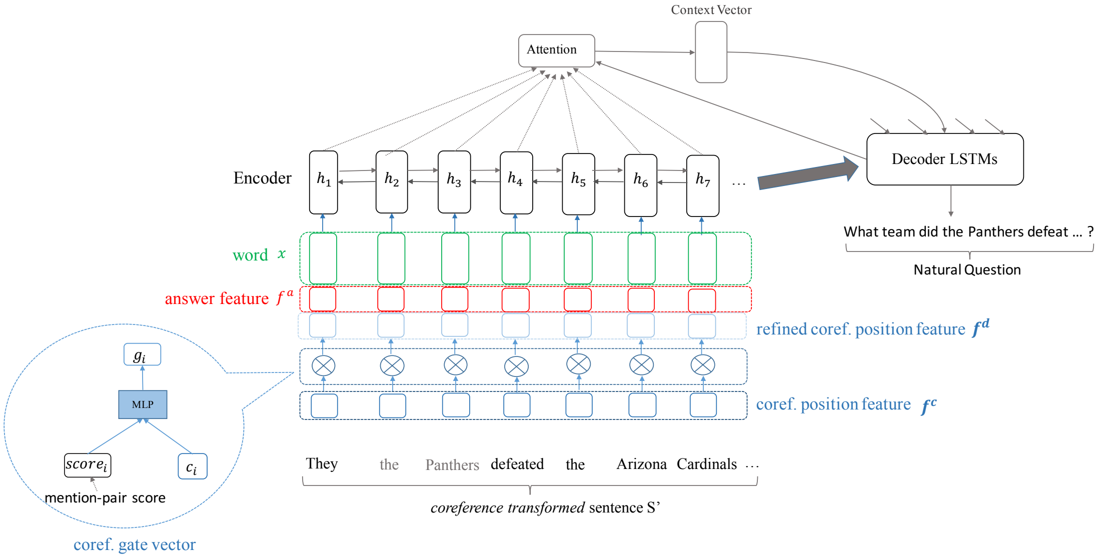

Context Vector
Attention
Decoder LSTMs
Encoder
h_1
h_2
h_3
h_4
h_5
h_6
h_7
...
What team did the Panthers defeat ... ?
Natural Question
word x
answer feature f^a
refined coref. position feature $\boldsymbol{f^d}$
coref. position feature $\boldsymbol{f^c}$
g_i
MLP
$score_i$
c_i
mention-pair score
coref. gate vector
They the Panthers defeated the Arizona Cardinals ...
coreference transformed sentence S'

Figure 6.2: The gated **Coref**erence knowledge for **N**eural **Q**uestion **G**eneration (**CorefNQG**) Model.

As shown in Figure 6.2, our generator prepares the feature-rich input embedding — a concatenation of (a) a refined coreference position feature embedding, (b) an answer feature embedding, and (c) a word embedding, each of which is described below. It then encodes the textual input using an LSTM unit [Hochreiter and Schmidhuber, 1997]. Finally, an attention-copy equipped decoder is used to decode the question.

More specifically, given the input sentence S (containing an answer span) and the preceding context C, we first run a coreference resolution system to get the coref-clusters for S and C and use them to create a *coreference transformed* input sentence: for each pronoun, we append its most representative non-pronominal coreferent mention. Specifically, we apply the simple feedforward network based mention-ranking model of Clark and Manning [2016] to the concatenation of C and S to get the coref-clusters for all entities in C and S. The C&M model produces a score/representation s for each mention pair (m_1, m_2),

$$s(m_1, m_2) = \mathbf{W}_m h_m(m_1, m_2) + b_m \tag{6.2}$$

where $\mathbf{W}_m$ is a $1 \times d$ weight matrix and b is the bias. $h_m(m_1, m_2)$ is representation of the last hidden layer of the three layer feedforward neural network.

For each pronoun in S, we then heuristically identify the most "representative" antecedent from its coref-cluster (proper nouns are preferred.) We append the new mention after the pronoun. For example, in Table 6.1, "the panthers" is the most representative mention in the coref-cluster for "they". The new sentence with the appended coreferent mention is our *coreference transformed* input sentence $S^{'}$ (see Figure 6.2).

word	they	the	panthers	defeated	the	arizona	cardinals	49	–	15	...
ans. fea.	O	O	O	O	B_ANS	I_ANS	I_ANS	O	O	O	...
coref. fea.	B_PRO	B_ANT	I_ANT	O	O	O	O	O	O	O	...

Table 6.1: Example input sentence with coreference and answer position features. The corresponding gold question is "What team did the Panthers defeat in the NFC championship game ?"

Coreference Position Feature Embedding For each token in $S^{'}$, we also maintain one position feature $\mathbf{f^c} = (c_1, ..., c_n)$, to denote pronouns (e.g., "they") and antecedents (e.g., "the panthers"). We use the BIO tagging scheme to label the associated spans in $S^{'}$. "B_ANT" denotes the start of an antecedent span, tag "I_ANT" continues the antecedent span and tag "O" marks tokens that do not form part of a mention span. Similarly, tags "B_PRO" and "I_PRO" denote the pronoun span. (See Table 6.1, "coref. feature".)

Refined Coref. Position Feature Embedding Inspired by the success of gating mechanisms for controlling information flow in neural networks [Hochreiter and Schmidhuber, 1997; Dauphin et al., 2017], we propose to use a gating network here to obtain a refined representation of the coreference position feature vectors $\mathbf{f^c} = (c_1, ..., c_n)$. The main idea is to utilize the mention-pair score (see Equation 6.2) to help the neural network learn the importance of the coreferent phrases. We compute the refined (gated) coreference position feature vector $\mathbf{f^d} = (d_1, ..., d_n)$ as follows,

$$\begin{aligned} g_i &= \text{ReLU}(\mathbf{W}_a c_i + \mathbf{W}_b score_i + b) \\ d_i &= g_i \odot c_i \end{aligned} \tag{6.3}$$

where $\odot$ denotes an element-wise product between two vectors and ReLU is the rectified linear activation function. $score_i$ denotes the mention-pair score for each antecedent token (e.g., “the” and “panthers”) with the pronoun (e.g., “they”); $score_i$ is obtained from the trained model (Equation 6.2) of the C&M. If token i is not added later as an antecedent token, $score_i$ is set to zero. $\mathbf{W}_a$, $\mathbf{W}_b$ are weight matrices and b is the bias vector.

Answer Feature Embedding We also include an answer position feature embedding to generate answer-specific questions; we denote the answer span with the usual BIO tagging scheme (see, e.g., “the arizona cardinals” in Table 6.1). During training and testing, the answer span feature (i.e., “`B_ANS`”, “`I_ANS`” or “`O`”) is mapped to its feature embedding space: $\mathbf{f^a} = (a_1, ..., a_n)$.

Word Embedding To obtain the word embedding for the tokens themselves, we just map the tokens to the word embedding space: $\mathbf{x} = (x_1, ..., x_n)$.

Final Encoder Input As noted above, the final input to the LSTM-based encoder is a concatenation of (1) the refined coreference position feature embedding (light blue units in Figure 6.2), (2) the answer position feature embedding (red units), and (3) the word embedding for the token (green units),

$$e_i = \text{concat}(d_i, a_i, x_i) \tag{6.4}$$

Encoder As for the encoder itself, we use bidirectional LSTMs to read the input $\mathbf{e} = (e_1, ..., e_n)$ in both the forward and backward directions. After encoding, we obtain two sequences of hidden vectors, namely, $\overrightarrow{\mathbf{h}} = (\overrightarrow{h_1}, ..., \overrightarrow{h_n})$ and $\overleftarrow{\mathbf{h}} = (\overleftarrow{h_1}, ..., \overleftarrow{h_n})$. The final output state of the encoder is the concatenation of $\overrightarrow{\mathbf{h}}$ and $\overleftarrow{\mathbf{h}}$ where

$$h_i = \text{concat}(\overrightarrow{h_i}, \overleftarrow{h_i}) \tag{6.5}$$

Question Decoder with Attention & Copy On top of the feature-rich encoder, we use LSTMs with attention [Bahdanau et al., 2015] as the decoder for generating the question $y_1, ..., y_m$ one token at a time. To deal with rare/unknown words, the decoder also allows directly copying words from the source sentence via pointing [Vinyals et al., 2015].

At each time step t, the decoder LSTM reads the previous word embedding w_{t-1} and previous hidden state s_{t-1} to compute the new hidden state,

$$s_t = \text{LSTM}(w_{t-1}, s_{t-1}) \tag{6.6}$$

Then we calculate the *attention distribution* α_t as in Bahdanau et al. [2015],

$$\begin{aligned} e_{t,i} &= h_i^T \mathbf{W}_c s_{t-1} \\ \alpha_t &= \text{softmax}(e_t) \end{aligned} \tag{6.7}$$

where $\mathbf{W}_c$ is a weight matrix and attention distribution α_t is a probability distribution over the source sentence words. With α_t, we can obtain the context vector h_t^*,

$$h_t^* = \sum_{i=1}^{n} \alpha_t^i h_i \tag{6.8}$$

Then, using the context vector h_t^* and hidden state s_t, the probability distribution over the target (question) side vocabulary is calculated as,

$$P_{vocab} = \text{softmax}(\mathbf{W}_d \text{concat}(h_t^*, s_t)) \tag{6.9}$$

Instead of directly using P_{vocab} for training/generating with the fixed target side vocabulary, we also consider *copying* from the source sentence. The copy probability is based on the context vector h_t^* and hidden state s_t,

$$\lambda_t^{copy} = \sigma\left(\mathbf{W}_e h_t^* + \mathbf{W}_f s_t\right) \tag{6.10}$$

and the probability distribution over the source sentence words is the sum of the attention scores of the corresponding words,

$$P_{copy}(w) = \sum_{i=1}^{n} \alpha_t^i * \mathbb{1}\{w == w_i\} \tag{6.11}$$

Finally, we obtain the probability distribution over the dynamic vocabulary (i.e., union of original target side and source sentence vocabulary) by summing over P_{copy} and P_{vocab},

$$P(w) = \lambda_t^{copy} P_{copy}(w) + (1 - \lambda_t^{copy}) P_{vocab}(w) \tag{6.12}$$

where σ is the sigmoid function, and $\mathbf{W}_d$, $\mathbf{W}_e$, $\mathbf{W}_f$ are weight matrices.

6.3.2 Answer Span Identification

We frame the problem of identifying candidate answer spans from a paragraph as a sequence labeling task and base our model on the BiLSTM-CRF approach for named entity recognition [Huang et al., 2015]. Given a paragraph of n tokens, instead of directly feeding the sequence of word vectors $\mathbf{x} = (x_1, ..., x_n)$ to the LSTM units, we first construct the feature-rich embedding $\mathbf{x}'$ for each token, which is the concatenation of the word embedding, an NER feature embedding, and a character-level representation of the word [Lample et al., 2016]. We use the concatenated vector as the "final" embedding $\mathbf{x}^{'}$ for the token,

$$x_i^{'} = \text{concat}(x_i, \text{CharRep}_i, \text{NER}_i) \tag{6.13}$$

where CharRep_i is the concatenation of the last hidden states of a character-based biLSTM. The intuition behind the use of NER features is that SQuAD answer spans contain a large number of named entities, numeric phrases, etc.

Then a multi-layer Bi-directional LSTM is applied to $(x_1^{'}, ..., x_n^{'})$ and we obtain the output state z_t for time step t by concatenation of the hidden states (forward and backward) at time step t from the last layer of the BiLSTM. We apply the softmax to $(z_1, ..., z_n)$ to get the normalized score representation for each token, which is of size $n \times k$, where k is the number of tags.

Instead of using a softmax training objective that minimizes the cross-entropy loss for each individual word, the model is trained with a CRF [Lafferty et al., 2001] objective, which minimizes the negative log-likelihood for the entire correct sequence: $-\log(p_{\mathbf{y}})$,

$$p_{\mathbf{y}} = \frac{\exp(q(\mathbf{x}^{'}, \mathbf{y}))}{\sum_{\mathbf{y}' \in \mathbf{Y}'} \exp(q(\mathbf{x}', \mathbf{y}'))} \tag{6.14}$$

where $q(\mathbf{x}^{'}, \mathbf{y}) = \sum_{t=1}^{n} P_{t,y_t} + \sum_{t=0}^{n-1} A_{y_t,y_{t+1}}$, P_{t,y_t} is the score of assigning tag y_t to the t^{th} token, and $A_{y_t,y_{t+1}}$ is the transition score from tag y_t to y_{t+1}, the scoring matrix A is to be learned. $\mathbf{Y}^{'}$ represents all the possible tagging sequences.

6.4 Experiments

6.4.1 Dataset

We use the SQuAD dataset [Rajpurkar et al., 2016] to train our models. It is one of the largest general purpose QA datasets derived from Wikipedia with over 100k questions posed by crowdworkers on a set of Wikipedia articles. The answer to each question is a segment of text from the corresponding Wiki passage. The crowdworkers were users of Amazon's Mechanical Turk located in the US or Canada. To obtain high-quality articles, the authors sampled 500 articles from the top 10,000 articles obtained by Nayuki's Wikipedia's internal PageRanks. The question-answer pairs were generated by annotators from a paragraph; and although the dataset is typically used to evaluate reading comprehension, it has also been used in an open domain QA setting [Chen et al., 2017; Wang et al., 2018]. For training/testing answer extraction systems, we pair each paragraph in the dataset with the gold answer spans that it contains. For the question generation system, we pair each sentence that contains an answer span with the corresponding gold question as in Du et al. [2017].

To quantify the effect of using predicted (rather than gold standard) answer spans on question generation (e.g., predicted answer span boundaries can be inaccurate), we also train the models on an augmented "Training set w/ noisy examples" (see Table 6.2). This training set contains all of the original training examples *plus* new examples for predicted answer spans (from the top-performing answer extraction model, bottom row of Table 6.4) that *overlap* with a gold answer span. We pair the new training sentence (w/ predicted answer span) with the gold question. The added examples comprise 42.21% of the noisy example training set.

For generation of our one million QA pair corpus, we apply our systems to the 10,000 top-ranking articles of Wikipedia.

6.4.2 Evaluation Metrics

For question generation evaluation, we use BLEU [Papineni et al., 2002] and METEOR [Denkowski and Lavie, 2014].[1] BLEU measures average n-gram precision vs. a set of reference questions and penalizes for overly short sentences. METEOR is a recall-oriented metric that takes into account synonyms, stemming, and paraphrases.

For answer candidate extraction evaluation, we use precision, recall and F-measure and compare with the gold standard SQuAD answers. Since answer boundaries are sometimes ambiguous, we compute *Binary Overlap* and *Proportional Overlap* metrics in addition to *Exact Match*. Binary Overlap counts every predicted answer that overlaps with a gold answer span as correct, and Proportional Overlap give partial credit proportional to the amount of overlap [Johansson and Moschitti, 2010; Irsoy and Cardie, 2014].

6.4.3 Baselines and Ablation Tests

For question generation, we compare to the state-of-the-art baselines and conduct ablation tests as follows: **Du et al. [2017]**'s model is an attention-based RNN sequence-to-sequence neural network (without using the answer location information feature). **Seq2seq + copy$_{\textbf{w/ answer}}$** is the attention-based

[1] We use the evaluation scripts of Du et al. [2017].

sequence-to-sequence model augmented with a copy mechanism, with answer features concatenated with the word embeddings during encoding. **Seq2seq + copy**$_{\textbf{w/ full context + answer}}$ is the same model as the previous one, but we allow access to the full context (i.e., all the preceding sentences and the input sentence itself). We denote it as **ContextNQG** henceforth for simplicity. **CorefNQG** is the coreference-based model that we propose in this paper. **CorefNQG w/o gating** is an ablation test, the gating network is removed and the coreference position embedding is not refined. **CorefNQG w/o mention-pair score** is also an ablation test where all mention-pair $score_i$ are set to zero.

For answer span extraction, we conduct experiments to compare the performance of an off-the-shelf NER system and BiLSTM based systems.

6.5 Results and Analysis

6.5.1 Automatic Evaluation

Models	Training set			Training set w/ noisy examples		
	BLEU-3	BLEU-4	METEOR	BLEU-3	BLEU-4	METEOR
Baseline [Du et al., 2017] (w/o answer)	17.50	12.28	16.62	15.81	10.78	15.31
Seq2seq + copy (w/ answer)	20.01	14.31	18.50	19.61	13.96	18.19
ContextNQG: Seq2seq + copy (w/ full context + answer)	20.31	14.58	18.84	19.57	14.05	18.19
CorefNQG	**20.90**	**15.16**	**19.12**	**20.19**	**14.52**	18.59
w/o gating	20.68	14.84	18.98	20.08	14.40	**18.64**
w/o mention-pair score	20.56	14.75	18.85	19.73	14.13	18.38

Table 6.2: Evaluation results for question generation.

Table 6.2 shows the BLEU-$\{3, 4\}$ and METEOR scores of different models. Our CorefNQG outperforms the seq2seq baseline of Du et al. [2017] by a large margin. This shows that the copy mechanism, answer features and coreference resolution all aid question generation. In addition, CorefNQG outperforms both Seq2seq+Copy models significantly, whether or not they have access to the full context. This demonstrates that the coreference knowledge encoded with the gating network explicitly helps with the training and generation: it is more difficult for the neural sequence model to learn the coreference knowledge in a latent way (See input 1 in Figure 6.3 for an example.) Building end-to-end models that take into account coreference knowledge in a latent way is an interesting direction to explore. In the ablation tests, the performance drop of CorefNQG w/o gating shows that the gating network is playing an important role for getting *refined* coreference position feature embedding, which helps the model learn the importance of an antecedent. The performance drop of CorefNQG w/o mention-pair score shows the mention-pair score introduced from the external system [Clark and Manning, 2016] helps the neural network better encode coreference knowledge.

	BLEU-3	BLEU-4	METEOR
Seq2seq + copy (w/ ans.)	17.81	12.30	17.11
ContextNQG	18.05	12.53	17.33
CorefNQG	**18.46**	**12.96**	**17.58**

Table 6.3: Evaluation results for question generation on the portion that requires coreference knowledge (36.42% examples of the original test set).

To better understand the effect of coreference resolution, we also evaluate our model and the baseline models on just that portion of the test set that requires pronoun resolution (36.42% of the examples) and show the results in Table 6.3. The gaps of performance between our model and the baseline models are still significant. Besides, we see that all three systems' performance drop on this partial test set, which demonstrates the hardness of generating questions for the cases that require pronoun resolution (passage context).

Input 1: The elizabethan navigator, sir francis drake was born in the nearby town of tavistock and was the mayor of plymouth. ... . he died of dysentery in 1596 off the coast of puerto rico.
Human: In what year did Sir Francis Drake die ?
ContextNQG: When did he die ?
CorefNQG: When did sir francis drake die ?

Input 2: american idol is an american singing competition ... . it began airing on fox on june 11 , 2002, as an addition to the idols format based on the british series pop idol and has since become one of the most successful shows in the history of american television.
Human: When did american idol first air on tv ?
ContextNQG: When did fox begin airing ?
CorefNQG: When did american idol begin airing ?

Input 3: ... the a38 dual-carriageway runs from east to west across the north of the city . within the city it is designated as ‘ the parkway ’ and represents the boundary between the urban parts of the city and the generally more recent suburban areas .
Human: What is the a38 called inside the city ?
ContextNQG: What is another name for the city ?
CorefNQG: What is the city designated as ?

Figure 6.3: Example questions (with answers highlighted) generated by human annotators (ground truth questions), by our system CorefNQG, and by the Seq2seq+Copy model trained with full context (i.e., ContextNQG).

We also show in Table 6.2 the results of the QG models trained on the training set augmented with noisy examples with predicted answer spans. There is a consistent but acceptable drop for each model on this new training set, given the inaccuracy of predicted answer spans. We see that CorefNQG still outperforms the baseline models across all metrics.

Figure 6.3 provides sample output for input sentences that require contextual coreference knowledge. We see that ContextNQG fails in all cases; our model misses only the third example due to an error introduced by coreference resolution — the “city” and “it” are considered coreferent. We can also see that human-generated questions are more natural and varied in form with better paraphrasing.

Models	Precision			Recall			F-measure		
	Prop.	Bin.	Exact	Prop.	Bin.	Exact	Prop.	Bin.	Exact
NER	24.54	25.94	12.77	**58.20**	**67.66**	**38.52**	34.52	37.50	19.19
BiLSTM	43.54	45.08	22.97	28.43	35.99	18.87	34.40	40.03	20.71
BiLSTM w/ NER	44.35	46.02	25.33	33.30	40.81	23.32	38.04	43.26	24.29
BiLSTM-CRF w/ char	**49.35**	**51.92**	**38.58**	30.53	32.75	24.04	37.72	40.16	29.62
BiLSTM-CRF w/ char w/ NER	45.96	51.61	33.90	41.05	43.98	28.37	**43.37**	**47.49**	**30.89**

Table 6.4: Evaluation results of answer extraction systems.

	Grammaticality	Making Sense	Answerability	Avg. rank
ContextNQG	3.793	3.836	3.892	1.768
CorefNQG	3.804*	3.847**	3.895*	1.762
Human	**3.807**	**3.850**	**3.902**	**1.758**

Table 6.5: Human evaluation results for question generation. "Grammaticality", "Making Sense" and "Answerability" are rated on a 1–5 scale (5 for the best, see the supplementary materials for a detailed rating scheme), "Average rank" is rated on a 1–3 scale (1 for the most preferred, ties are allowed.)

In Table 6.4, we show the evaluation results for different answer extraction models. First we see that all variants of BiLSTM models outperform the off-the-shelf NER system (that proposes all NEs as answer spans), though the NER system has a higher recall. The BiLSTM-CRF that encodes the character-level and NER features for each token performs best in terms of F-measure.

6.5.2 Human Study

We hired four native speakers of English to rate the systems' outputs. Detailed guidelines for the raters are listed in the supplementary materials. The evaluation can also be seen as a measure of the quality of the generated dataset (Section 6.5.3). We randomly sampled 11 passages/paragraphs from the test set; there are in total around 70 question-answer pairs for evaluation.

We consider three metrics — "grammaticality", "making sense" and "answerability". The evaluators are asked to first rate the grammatical correctness of the generated question (before being shown the associated input sentence or any other textual context). Next, we ask them to rate the degree to which the question "makes sense" given the input sentence (i.e., without considering the correctness of the answer span). Finally, evaluators rate the "answerability" of the question given the full context.

Table 6.5 shows the results of the human evaluation.[2] Bold indicates top scores. We see that the original human questions are preferred over the two NQG systems' outputs, which is understandable given the examples in Figure 6.3. The human-generated questions make more sense and correspond better with the provided answers, particularly when they require information in the preceding context. How exactly to capture the preceding context so as to *ask* better and more diverse questions is an interesting future direction for research. In terms of grammaticality, however, the neural models do quite well, achieving very close to human performance. In addition, we see that our method (CorefNQG) performs statistically significantly better across all metrics in comparison to the baseline model (ContextNQG), which has access to the entire preceding context in the passage.

6.5.3 The Generated Corpus

Our system generates in total 1,259,691 question-answer pairs, nearly 126 questions per article. Figure 6.4 shows the distribution of different types of questions in our dataset vs. the SQuAD training set. We see that the distribution for "In what", "When", "How long", "Who", "Where", "What does" and "What do" questions

[2] Two-tailed t-test results are shown for our method compared to ContextNQG (statistical significance is indicated with $^*(p < 0.05)$, $^{**}(p < 0.01)$.)

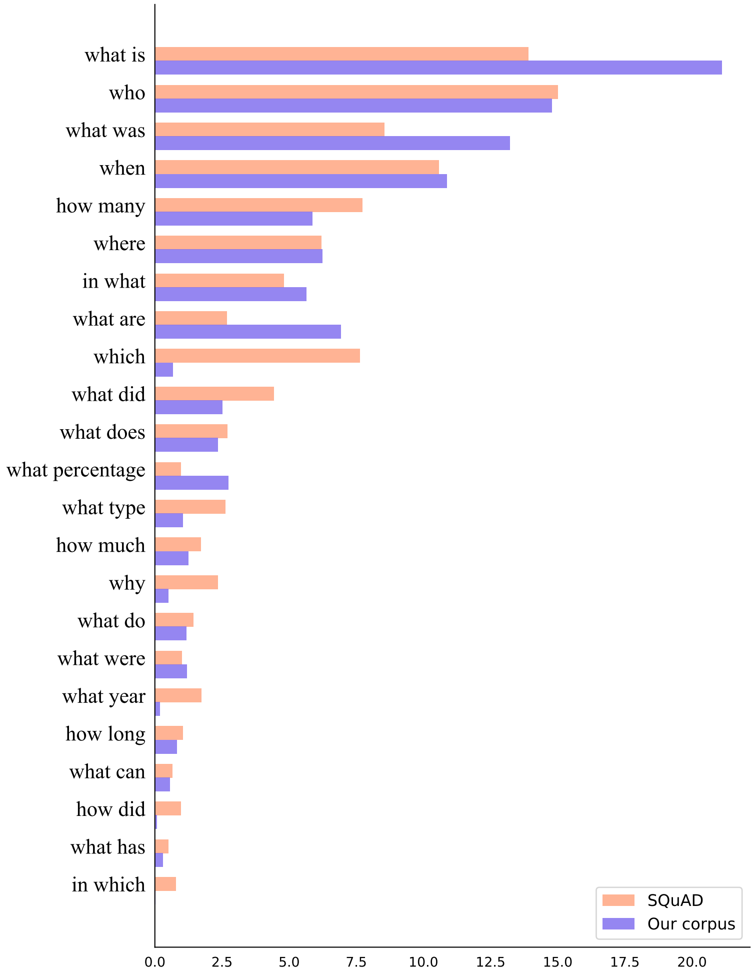


Figure 6.4: Distribution of question types of our corpus and SQuAD training set. The categories are the ones used in Wang et al. [2016], we add one more category: "what percentage".

in the two datasets is similar. Our system generates more "What is", "What was" and "What percentage" questions, while the proportions of "What did", "Why" and "Which" questions in SQuAD are larger than ours. One possible reason is that the "Why", "What did" questions are more *complicated* to ask (sometimes involving world knowledge) and the answer spans are longer phrases of various types that are harder to identify. "What is" and "What was" questions, on the other hand, are often *safer* for the neural networks systems to ask.

The United States of America (USA), commonly referred to as the United States (U.S.) or America, is a federal republic composed of states, a federal district, five major self-governing territories, and various possessions. ... . The territories are scattered about the Pacific Ocean and the Caribbean Sea. Nine time zones are covered. The geography, climate and wildlife of the country are extremely diverse.
Q1: What is another name for the united states of america ?
Q2: How many major territories are in the united states?
Q3: What are the territories scattered about ?

Figure 6.5: Example question-answer pairs from our generated corpus.

	Exact Match		F-1	
	Dev	Test	Dev	Test
DocReader [Chen et al., 2017]	82.33	81.65	88.20	87.79

Table 6.6: Performance of the neural machine reading comprehension model (no initialization with pretrained embeddings) on our generated corpus.

In Figure 6.5, we show some examples of the generated question-answer pairs. The answer extractor identifies the answer span boundary well and all three questions correspond to their answers. Q2 is valid but not entirely accurate.

Table 6.6 shows the performance of a top-performing system for the SQuAD dataset (Document Reader [Chen et al., 2017]) when applied to the development and test set portions of our generated dataset. The system was trained on the training set portion of our dataset. We use the SQuAD evaluation scripts, which calculate exact match (EM) and F-1 scores.[3] Performance of the neural machine reading model is reasonable. We also train the DocReader on our training set and test the models' performance on the *original* dev set of SQuAD; for this, the performance is around 45.2% on EM and 56.7% on F-1 metric. DocReader trained on the *original* SQuAD training set achieves 69.5% EM, 78.8% F-1 indicating that our dataset is more difficult and/or less natural than the crowd-sourced QA pairs

[3] F-1 measures the average overlap between the predicted answer span and ground truth answer [Rajpurkar et al., 2016].

of SQuAD.

6.6 Chapter Summary

To summarize this chapter, we introduce a new neural network model for better encoding coreference knowledge for paragraph-level generation of synthetic question-answer pairs. Evaluations with different metrics on the SQuAD machine reading dataset show that our model outperforms state-of-the-art baselines. The ablation study shows the effectiveness of different components in our model. Finally, we apply our question generation framework to produce a corpus of 1.26 million QA pairs, which we hope will benefit the QA research community.

CHAPTER 7
CONCLUSIONS AND FUTURE WORK

In this dissertation, we propose deep learning based models and frameworks for document-level information extraction. To better capture the document-level context and structure for understanding the document and conducting more coherent extractions, we propose two neural network based approaches (i.e., sequence labeling-based and neural generation-based models). The pre-trained language model representations enable to model to have a basic understanding of words (in the context of other words) and basic linguistic knowledge. To better access knowledge encoded in the pre-trained models, we formulate the event extraction problem as a question answering task and propose more semantically meaningful question generation strategies for the framework.

7.1 Summary of Contributions

In Chapter 3, we investigate how end-to-end neural sequence models (with pre-trained language model representations) perform on document-level role filler extraction, as well as how the length of context captured affects the models' performance. Then we propose a novel multi-granularity reader to dynamically aggregate information captured by neural representations learned at different levels of granularity (e.g., the sentence- and paragraph-level). In evaluations, we show that our best system performs substantially better than prior work which reads through the document sentence by sentence.

Chapter 4 introduces a generative transformer-based encoder-decoder framework (GRIT) and its extension GTT that are designed to model context at the document level, for the classic problem of template filling. More specifically, GRIT tackles the sub-task called role-filler entity extraction (REE) and GTT is capable of handling the full task. They can make extraction decisions across sentence boundaries and has the capacity to respect cross-role dependencies in the template structure. Plus, GTT is better at capturing the dependencies across multiple events. We demonstrate that our models perform substantially better than prior works which are mostly sequence labeling based and conduct the extraction for each template independently.

To mitigate the problem of error propagation in event extraction (i.e., from entity recognition to argument type assignment) and better exploit the relatedness between different argument role names, Chapter 5 introduces a new paradigm for event extraction by formulating it as a question answering (QA) task that extracts the event arguments in an end-to-end manner. Empirical results demonstrate that our framework outperforms prior methods substantially; in addition, it is capable of extracting event arguments for roles not seen at training time (i.e., in a zero-shot learning setting).

Motivated by the advantage of using QA formulation for IE, in Chapter 6 we propose a framework for generating additional synthetic QA pairs from Wikipedia articles. We propose a neural network approach that incorporates coreference knowledge via a novel gating mechanism. Compared to models that only take into account sentence-level information [Heilman and Smith, 2010; Du et al., 2017], we find that the linguistic knowledge introduced by the coreference representation aids question generation significantly, producing models that outperform the current state-of-the-art. We apply our system (composed of an answer span extraction system and the passage-level QG system) to the 10,000 top-ranking Wikipedia articles and create a corpus of over one million question0-answer pairs.

7.2 Future Horizons

I've summarized my contributions in this dissertation on making sense of long and unstructured documents. Next, I will explain my future plans on (1) increasing the reasoning capability and reducing the cost of building/applying the document machine reader; (2) how techniques that I introduce can achieve broader impact outside NLP (i.e., via interdisciplinary research).

Reasoning Capabilities for Machine Reader When reading documents and making decisions, humans rely on different forms of knowledge (such as linguistic, analogical, procedural and commonsense). But current end-to-end learning models rarely capture them. In my investigation on procedural passage understanding [Du et al., 2019] in collaboration with researchers from Allen Institute for AI, we found that end-to-end neural models often make inconsistent predictions across different passages about the same procedure. For example in photosynthesis, water is moved&evaporated and oxygen is created. While the specific descriptions for this process change, the neural passage reader is unable to make globally consistent predictions that include the three effects. I proposed to leverage consistency bias into the model during training, which proves to help improve performance and consistency.

I believe leveraging symbolic representations for language (e.g., building a consistency bias into the model for the example above) is key to building more robust and accurate document readers. On the reverse side, *acquiring* knowledge from text, e.g., commonsense knowledge regarding object size/weight and temporal frequency/duration is also worth investigating.

Efficient Modeling Requiring Lower Cost Designing resource-efficient algorithms is essential to make technologies more accessible to people who want to build customized machine readers for various domains. Plus, resource-efficient algorithms also help to achieve a balance between obtaining strong results and energy cost [Schwartz et al., 2019].

As input text/documents become longer, the computation time and memory usage of end-to-end models (e.g., transformer-based) grow exponentially. I plan to draw insights from *cognitive science* research on global reading strategies [Mokhtari and Reichard, 2002] and working memory [Gathercole and Baddeley, 2014] for the designing of efficient document reading algorithms. In real life, when reading very long documents (e.g., a chapter in a book), people don't tend to focus on *all* the words or remember all the details at the same time; instead, reading paragraph by paragraph and only referring to the previous context that is related is easier. Based on this observation, from the methodology's perspective, I plan to also learn from the information retrieval community, to design a more efficient, interpretable, and accurate machine reader that *retrieves* relevant context from the long document when needed.

Also, for the task of automatic question generation, humans ask richer, more informative, and creative questions in a much more efficient way, than current end-to-end systems [Rothe et al., 2017], which rely on a huge number of sentence-question examples for training. I believe drawing insights from how and when human asks questions (e.g., ask to learn and seek information) is essential for machines to ask more informative and novel questions with fewer examples.

Apart from drawing insights from cognitive science, from the hardware's perspective, how to design specialized efficient document readers for different hardware platforms is also interesting to investigate.

Broader Impact of NLP in Interdisciplinary Research With tons of articles written (online and offline) in various domains (e.g., news articles, scientific papers, proprietary documents), efficient text understanding and processing are becoming more and more important, for bridging the gap between fast-growing text data and people's limited information processing capability. To achieve a broader impact outside NLP and ML — by applying my research in domain specific contexts, I believe advanced applications that tackle customized needs from different types of audiences can be created. To name a few representative areas:

– *Understanding Scientific Literature.*

Designing NLP techniques for processing textual information in scientific literature is a challenging but meaningful task. They can help to efficiently understand research contributions and methodology in the research work. I'm interested in investigating how to build document extraction tools for extracting structured information from scientific papers. For example, to understand the methodology proposed in one paper, extracting entities and their relations is a preliminary step; to understand relationships between different papers, cross-document extraction for citation analysis would be important.

However, language usage varies significantly across research communities, and the specific researcher will have different needs for the structured outputs. Thus, my goal is to design accurate domain-specific document readers, that can take into consideration user-specific needs and user feedback.

– *Understanding Proprietary Documents.*

Privacy policies and similar proprietary documents are long and complex documents that are difficult for users to read. But they have a legal influence on many aspects of user interests (e.g., user data). Applications like extracting salient items from legal documents and answering user-specific questions are of general interest. Challenges such as the high cost for obtaining expert-level annotations further motivate NLP research in data collection methodology, as well as data-efficient algorithms.

Overall from another perspective, problems met when conducting domain-specific research and building applications can also help evoke methodology innovations in my research on NLP.

BIBLIOGRAPHY

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Berg-Kirkpatrick, T., Burkett, D., and Klein, D. (2012). An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

Chambers, N. (2013). Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Seattle, Washington, USA. Association for Computational Linguistics.

Chambers, N. and Jurafsky, D. (2011). Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 976–986, Portland, Oregon, USA. Association for Computational Linguistics.

Chen, D., Bolton, J., and Manning, C. D. (2016). A thorough examination of the cnn/daily mail reading comprehension task. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany. Association for Computational Linguistics.

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of* *the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879. Association for Computational Linguistics.

Chen, Y., Xu, L., Liu, K., Zeng, D., and Zhao, J. (2015). Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Cheung, J. C. K., Poon, H., and Vanderwende, L. (2013). Probabilistic frame induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 837–846, Atlanta, Georgia. Association for Computational Linguistics.

Chinchor, N. (1992). Muc-4 evaluation metrics. In *MUC*.

Chinchor, N., Lewis, D. D., and Hirschman, L. (1993). Evaluating message understanding systems: an analysis of the third message understanding conference (muc-3). Technical report, SCIENCE APPLICATIONS INTERNATIONAL CORP SAN DIEGO CA.

Chiu, J. P. and Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Ciaramita, M. and Altun, Y. (2006). Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602.

Clark, K. and Manning, C. D. (2016). Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653. Association for Computational Linguistics.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. In *International Conference on Machine Learning*, pages 933–941.

Denkowski, M. and Lavie, A. (2014). Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380, Baltimore, Maryland, USA. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., and

Weischedel, R. (2004). The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Dong, L., Yang, N., Wang, W., Wei, F., Liu, X., Wang, Y., Gao, J., Zhou, M., and Hon, H.-W. (2019). Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13042–13054.

Du, X. and Cardie, C. (2017). Identifying where to focus in reading comprehension for neural question generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2067–2073. Association for Computational Linguistics.

Du, X. and Cardie, C. (2020). Document-level event role filler extraction using multi-granularity contextualized encoding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8010–8020, Online. Association for Computational Linguistics.

Du, X., Dalvi, B., Tandon, N., Bosselut, A., Yih, W.-t., Clark, P., and Cardie, C. (2019). Be consistent! improving procedural text comprehension using label consistency. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Du, X., Rush, A., and Cardie, C. (2021a). GRIT: Generative role-filler transformers for document-level event entity extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 634–644, Online. Association for Computational Linguistics.

Du, X., Rush, A., and Cardie, C. (2021b). Template filling with generative transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 909–914, Online. Association for Computational Linguistics.

Du, X., Shao, J., and Cardie, C. (2017). Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352. Association for Computational Linguistics.

Duan, N., Tang, D., Chen, P., and Zhou, M. (2017a). Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874. Association for Computational Linguistics.

Duan, S., He, R., and Zhao, W. (2017b). Exploiting document level information to improve event detection via recurrent neural networks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing*, pages 352–361, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Ebner, S., Xia, P., Culkin, R., Rawlins, K., and Van Durme, B. (2020). Multi-sentence argument linking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8057–8077, Online. Association for Computational Linguistics.

Feng, X., Qin, B., and Liu, T. (2018). A language-independent neural network for event detection. *Science China Information Sciences*, 61(9):092106.

FitzGerald, N., Michael, J., He, L., and Zettlemoyer, L. (2018). Large-scale QA-SRL parsing. In *Proceedings of the 56th Annual Meeting of the Association* *for Computational Linguistics (Volume 1: Long Papers)*, pages 2051–2060, Melbourne, Australia. Association for Computational Linguistics.

Gathercole, S. E. and Baddeley, A. D. (2014). *Working memory and language*. Psychology Press.

Gers, F. A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: Continual prediction with lstm. *Neural Comput.*

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Grishman, R. (2012). Information extraction: Capabilities and challenges. *Notes prepared*.

Grishman, R. (2019). Twenty-five years of information extraction. *Natural Language Engineering*, 25(6):677–692.

Grishman, R. and Sundheim, B. (1996). Design of the MUC-6 evaluation. In *TIPSTER TEXT PROGRAM PHASE II: Proceedings of a Workshop held at Vienna, Virginia, May 6-8, 1996*, pages 413–422, Vienna, Virginia, USA. Association for Computational Linguistics.

He, T., Tan, X., Xia, Y., He, D., Qin, T., Chen, Z., and Liu, T.-Y. (2018). Layer-wise coordination between encoder and decoder for neural machine translation. In *Advances in Neural Information Processing Systems*, pages 7944–7954.

Heilman, M. (2011). *Automatic factual question generation from text*. PhD thesis, Ph. D. thesis, Carnegie Mellon University.

Heilman, M. and Smith, N. A. (2010). Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Con-*

ference of the North American Chapter of the Association for Computational Linguistics, pages 609–617, Los Angeles, California. Association for Computational Linguistics.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701.

Hill, F., Bordes, A., Chopra, S., and Weston, J. (2015). The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Huang, L., Ji, H., Cho, K., Dagan, I., Riedel, S., and Voss, C. (2018). Zero-shot transfer learning for event extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia. Association for Computational Linguistics.

Huang, R. and Riloff, E. (2011). Peeling back the layers: Detecting event role fillers in secondary contexts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1137–1147, Portland, Oregon, USA. Association for Computational Linguistics.

Huang, R. and Riloff, E. (2012). Modeling textual cohesion for event extraction. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Huang, Z., Xu, W., and Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *ArXiv*, abs/1508.01991.

Irsoy, O. and Cardie, C. (2014). Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 720–728. Association for Computational Linguistics.

Jain, S., van Zuylen, M., Hajishirzi, H., and Beltagy, I. (2020). SciREX: A challenge dataset for document-level information extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7506–7516, Online. Association for Computational Linguistics.

Ji, H. and Grishman, R. (2008). Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262, Columbus, Ohio. Association for Computational Linguistics.

Jia, R., Wong, C., and Poon, H. (2019). Document-level n-ary relation extraction with multiscale representation learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3693–3704, Minneapolis, Minnesota. Association for Computational Linguistics.

Johansson, R. and Moschitti, A. (2010). Syntactic and semantic structure for opinion expression detection. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 67–76. Association for Computational Linguistics.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. (2017). TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Joshi, M., Levy, O., Weld, D. S., and Zettlemoyer, L. (2019). Bert for coreference resolution: Baselines and analysis. *arXiv preprint arXiv:1908.09091*.

Jurafsky, D. and Martin, J. H. (2014). *Speech and language processing*. Prentice Hall, Pearson Education International.

Katiyar, A. and Cardie, C. (2018). Nested named entity recognition revisited. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 861–871, New Orleans, Louisiana. Association for Computational Linguistics.

Kim, J.-D., Ohta, T., Tateisi, Y., and Tsujii, J. (2003). Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. (2019). Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Labutov, I., Basu, S., and Vanderwende, L. (2015). Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 889–898.

Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Lample, G., Ott, M., Conneau, A., Denoyer, L., and Ranzato, M. (2018). Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium. Association for Computational Linguistics.

Lee, K., He, L., Lewis, M., and Zettlemoyer, L. (2017). End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Levesque, H. J., Davis, E., and Morgenstern, L. (2011). The winograd schema challenge. In *Aaai spring symposium: Logical formalizations of commonsense reasoning*, volume 46, page 47.

Levy, O., Seo, M., Choi, E., and Zettlemoyer, L. (2017). Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Li, Q., Ji, H., and Huang, L. (2013). Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 73–82, Sofia, Bulgaria. Association for Computational Linguistics.

Li, S., Ji, H., and Han, J. (2021). Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, Online. Association for Computational Linguistics.

Li, X., Feng, J., Meng, Y., Han, Q., Wu, F., and Li, J. (2019a). A unified mrc framework for named entity recognition. *arXiv preprint arXiv:1910.11476*.

Li, X., Nguyen, T. H., Cao, K., and Grishman, R. (2015). Improving event detection with abstract meaning representation. In *Proceedings of the First Workshop on Computing News Storylines*, pages 11–15, Beijing, China. Association for Computational Linguistics.

Li, X., Yin, F., Sun, Z., Li, X., Yuan, A., Chai, D., Zhou, M., and Li, J. (2019b). Entity-relation extraction as multi-turn question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1340–1350, Florence, Italy. Association for Computational Linguistics.

Liao, S. and Grishman, R. (2010). Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797, Uppsala, Sweden. Association for Computational Linguistics.

Lin, Y., Ji, H., Huang, F., and Wu, L. (2020). A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of* *the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Linguistic Data Consortium, L. (2005). English annotation guidelines for events. *https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf.*

Liu, J., Chen, Y., Liu, K., Bi, W., and Liu, X. (2020). Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651, Online. Association for Computational Linguistics.

Liu, S., Chen, Y., Liu, K., and Zhao, J. (2017). Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1789–1798, Vancouver, Canada. Association for Computational Linguistics.

Liu, X., Huang, H., and Zhang, Y. (2019). Open domain event extraction using neural latent variable models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2860–2871, Florence, Italy. Association for Computational Linguistics.

Liu, X., Luo, Z., and Huang, H. (2018). Jointly multiple events extraction via attention-based graph information aggregation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1247–1256, Brussels, Belgium. Association for Computational Linguistics.

Luan, Y., He, L., Ostendorf, M., and Hajishirzi, H. (2018). Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in*

Natural Language Processing, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Luan, Y., Wadden, D., He, L., Shah, A., Ostendorf, M., and Hajishirzi, H. (2019). A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3036–3046, Minneapolis, Minnesota. Association for Computational Linguistics.

Luo, X. (2005). On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Ma, X. and Hovy, E. (2016). End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

McCann, B., Keskar, N. S., Xiong, C., and Socher, R. (2018). The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Miller, A., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A., and Weston, J. (2016). Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409. Association for Computational Linguistics.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Mitkov, R. and Ha, L. A. (2003). Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing-Volume 2*, pages 17–22. Association for Computational Linguistics.

Mokhtari, K. and Reichard, C. A. (2002). Assessing students' metacognitive awareness of reading strategies. *Journal of educational psychology.*

MUC-4 (1992). Fourth message understanding conference (MUC-4). In *Proceedings of FOURTH MESSAGE UNDERSTANDING CONFERENCE (MUC-4)*, McLean, Virginia.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.

Nguyen, T. H., Cho, K., and Grishman, R. (2016). Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies, pages 300–309, San Diego, California. Association for Computational Linguistics.

Nguyen, T. H. and Grishman, R. (2015). Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 365–371, Beijing, China. Association for Computational Linguistics.

Nguyen, T. M. and Nguyen, T. H. (2019). One for all: Neural joint modeling of entities and events. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6851–6858.

Okurowski, M. E. (1993). Information extraction overview. Technical report, NATIONAL COMPUTER SECURITY CENTER FORT GEORGE G MEADE MD.

Olney, A. M., Graesser, A. C., and Person, N. K. (2012). Question generation from concept maps. *Dialogue & Discourse*, 3(2):75–99.

Paolini, G., Athiwaratkun, B., Krone, J., Ma, J., Achille, A., Anubhai, R., dos Santos, C. N., Xiang, B., and Soatto, S. (2021). Structured prediction as translation between augmented natural languages. In *9th International Conference on Learning Representations, ICLR 2021*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Patwardhan, S. and Riloff, E. (2009). A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 151–160, Singapore. Association for Computational Linguistics.

Peng, N., Poon, H., Quirk, C., Toutanova, K., and Yih, W.-t. (2017). Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Peters, M. E., Ruder, S., and Smith, N. A. (2019). To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.

Quirk, C. and Poon, H. (2017). Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1171–1182, Valencia, Spain. Association for Computational Linguistics.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *https://s3-us-west-2.amazonaws. com/openai-assets/researchcovers/languageunsupervised/languageunderstandingpaper.pdf.*

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683.*

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Reddy, S., Raghu, D., Khapra, M. M., and Joshi, S. (2017). Generating natural language question-answer pairs from a knowledge graph using a rnn based question generation model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 376–385. Association for Computational Linguistics.

Rothe, A., Lake, B. M., and Gureckis, T. (2017). Question asking as program generation. In *Advances in neural information processing systems.*

Rus, V., Wyse, B., Piwek, P., Lintean, M., Stoyanchev, S., and Moldovan, C. (2010). The first question generation shared task evaluation challenge. In *Pro-*

ceedings of the 6th International Natural Language Generation Conference, pages 251–257. Association for Computational Linguistics.

Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. (2019). Green ai. *arXiv preprint arXiv:1907.10597*.

Serban, I. V., García-Durán, A., Gulcehre, C., Ahn, S., Chandar, S., Courville, A., and Bengio, Y. (2016). Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598, Berlin, Germany. Association for Computational Linguistics.

Sha, L., Qian, F., Chang, B., and Sui, Z. (2018). Jointly extracting event triggers and arguments by dependency-bridge rnn and tensor-based argument interaction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Spolaor, N., Cherman, E. A., Monard, M. C., and Lee, H. D. (2013). A comparison of multi-label feature selection methods using the problem transformation approach. *Electronic Notes in Theoretical Computer Science*, 292:135–151.

Sundheim, B. M. (1991). Overview of the third Message Understanding Evaluation and Conference. In *Third Message Uunderstanding Conference (MUC-3): Proceedings of a Conference Held in San Diego, California, May 21-23, 1991*.

Sundheim, B. M. (1992). Overview of the fourth message understanding evaluation and conference. In *FOURTH MESSAGE UNDERSTANDING CONFERENCE (MUC-4), Proceedings of a Conference Held in McLean, Virginia, June 16-18, 1992*.

Sundheim, B. M. (1993). The Message Understanding Conferences. In *TIPSTER TEXT PROGRAM: PHASE I: Proceedings of a Workshop held at Fredricksburg, Virginia, September 19-23, 1993*, pages 5–5, Fredericksburg, Virginia, USA. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Trinh, T., Dai, A., Luong, T., and Le, Q. (2018). Learning longer-term dependencies in RNNs with auxiliary losses. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4965–4974, Stockholmsmässan, Stockholm Sweden. PMLR.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017a). Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017b). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Wadden, D., Wennberg, U., Luan, Y., and Hajishirzi, H. (2019). Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China. Association for Computational Linguistics.

Walker, C., Strassel, S., Medero, J., and Maeda, K. (2006). Ace 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 57.

Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., Zhang, W., Chang, S., Tesauro, G., Zhou, B., and Jiang, J. (2018). R3: Reinforced ranker-reader for open-domain question answering. In *AAAI*.

Wang, Z., Mi, H., Hamza, W., and Florian, R. (2016). Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211*.

Winograd, T. (1972). Understanding natural language. *Cognitive psychology*, 3(1):1–191.

Yang, B. and Mitchell, T. M. (2016). Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, California. Association for Computational Linguistics.

Yao, X., Bouma, G., and Zhang, Y. (2012). Semantics-based question generation and implementation. *Dialogue & Discourse*, 3(2):11–42.

Yao, Y., Ye, D., Li, P., Han, X., Lin, Y., Liu, Z., Liu, Z., Huang, L., Zhou, J., and Sun, M. (2019). DocRED: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Annual Meeting of the Association for*

Computational Linguistics, pages 764–777, Florence, Italy. Association for Computational Linguistics.

Zhang, J., Qin, Y., Zhang, Y., Liu, M., and Ji, D. (2019a). Extracting entities and events as a single task using a transition-based neural model. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5422–5428. AAAI Press.

Zhang, T., Ji, H., and Sil, A. (2019b). Joint entity and event extraction with generative adversarial imitation learning. *Data Intelligence*, 1(2):99–120.

Zhang, Y. and Clark, S. (2011). Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 37(1):105–151.

Zhang, Y., Zhong, V., Chen, D., Angeli, G., and Manning, C. D. (2017). Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

Zhao, Y., Jin, X., Wang, Y., and Cheng, X. (2018). Document embedding enhanced event detection with hierarchical and supervised attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 414–419, Melbourne, Australia. Association for Computational Linguistics.

Zheng, S., Cao, W., Xu, W., and Bian, J. (2019). Doc2edag: An end-to-end document-level framework for chinese financial event extraction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 337–346.

Zhou, Q., Yang, N., Wei, F., Tan, C., Bao, H., and Zhou, M. (2017). Neural question generation from text: A preliminary study. In *Natural Language Processing and Chinese Computing - 6th CCF International Conference, NLPCC 2017, Dalian, China, November 8-12, 2017, Proceedings*, volume 10619, pages 662–671. Springer.