The Dissertation Committee for Pengxiang Cheng
certifies that this is the approved version of the following dissertation:

Learning Better Latent Representations from Semantic Knowledge

Committee:

Katrin Erk, Supervisor

Raymond Mooney

Gregory Durrett

Hannaneh Hajishirzi

Learning Better Latent Representations from Semantic Knowledge

by

Pengxiang Cheng

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2020

Acknowledgments

It is hard for me to believe how the past seven years of my doctoral studies have come to an end without the chance to meet in person and express my gratitude to so many wonderful mentors, friends, and family members. Without their wholehearted love and support, this research and this dissertation would not have been possible.

First and foremost, I would like to thank my advisor Katrin Erk for her guidance and support throughout the years. She spent countless hours to inspire me to think broader and dig deeper, to share with me both intellectual and personal experience, to encourage me through difficult times, and to lead me via her meticulous and perfectionist attitude toward science. I am honored to have worked with her, and I am genuinely grateful.

I would like to thank Ray Mooney, Greg Durrett, and Hanna Hajishirzi, for serving on my committee and providing valuable comments and feedback on my dissertation. I also want to thank Jessy Li for stimulating many insightful discussions that helped me find new directions. It would be very remiss of me for not mentioning Dana Ballard, who guided me patiently during my first year here on studying computational muscle control of humanoid movement. I learned a great deal from him on how to approach complex problems and acquire knowledge from new domains. Dana continues to be a mentor and inspiration for me.

Additional thanks to many lab mates and fellow graduate students, and without their help and friendship, my graduate school journey would have never been the same. This set of people includes, but is not limited to, Ruohan Zhang, Leif Johnson, Stephen Roller, Alex Rosenfeld, Eric Holgate, Elisa Ferracane, Jiacheng Xu, Jifan Chen, and Yasumasa Onoe. Also, special thanks to my longtime friends, Peisen Zhao for being a drinking partner, and Zhirong Wu for providing phone counseling services.

Finally, I own a great debt to my parents, Yusheng Cheng and Xia Wang, for their unwavering support throughout my 28 years of life. I cannot thank them enough for their sacrifices, encouragement, and everything. And thanks to my girlfriend, Fangyu Zhang, without whose love my PhD grind would have become much harder.

Learning Better Latent Representations from Semantic Knowledge

Pengxiang Cheng, Ph.D.

The University of Texas at Austin, 2020

Supervisor: Katrin Erk

Many modern efforts in Natural Language Processing involve the use of deep neural network models, where dense vector representations are learned for words and sentences, and they have been proven effective in many downstream tasks. However, it remains unknown whether these representations truly understand the meaning of language, due to their vulnerability against adversarial attacks and lack of generalization ability to unseen domains.

In this thesis, we investigate the use of semantic knowledge to help learn better representations from neural models. We start with a certain type of semantic phenomenon, the implicit predicate-argument relations, and propose two neural models that draw on narrative event coherence and entity salience. We also introduce an argument cloze task for the automatic creation of synthetic data at scale from structural representations of events and entities. We demonstrate that when trained with large-scale synthetic data, both these models show good performance on a human-annotated dataset for nominal implicit arguments.

We then focus on the integration of a broader range of semantic knowledge into neural models in a more latent manner. We find that by injecting coreference knowledge as auxiliary supervision for self-attention, a relatively small model sets the state-of-the-art on a word prediction task specifically designed to require long-distance reasoning. We further explore different ways of integrating semantic knowledge into large-scale pre-trained language models to make them more generalizable at out-of-domain question answering tasks, and show some preliminary results.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent Natural Language Processing (NLP) research, we have seen substantial progress in a wide range of semantic tasks (for example, question answering and natural language inference), largely accredited to the development of powerful deep neural models and the fast-growing computing resources that facilitate the training of large models with massive data. Many of these approaches take the idea of end-to-end training, that is, to train a single neural network that takes the raw text in its natural language form as input and directly produces task-specific outputs (e.g., a label in a classification task, or a sequence of tokens in a generation task). However, current models often fail on problems where sophisticated reasoning of the semantic structure is explicitly needed. In this thesis, we seek to enhance the end-to-end training approach with different techniques for integrating semantic knowledge.

The trend of end-to-end training paradigm in NLP is accelerated by the introduction of pre-trained word embeddings (Mikolov et al., 2013, Pennington et al., 2014) where words are mapped to dense vectors of real numbers as their meaning representations, with the idea rooted from the hypothesis of distributional semantics: "A word is characterized by the company it keeps" (Firth, 1957). These pre-trained embeddings have been shown to greatly improve the performance in tasks like syntactic parsing (Chen and Manning, 2014), text classification (Kim, 2014), machine translation (Cho et al., 2014), named entity recognition (Lample et al., 2016), question answering (Seo et al., 2017), and so on. Word embeddings are adopted in place of the traditional feature engineering approach, with the expectation that the dense vector for each word would capture most of the hand-crafted features required for the end tasks.

There is one major limitation of the pre-trained word embeddings: They are context-independent, in that the embedding vector for a word does not change when the word is surrounded by different contexts. This is contradictory to the fact that word meanings in natural language are highly context-sensitive. Therefore, more recently we have seen a surge of pre-trained *contextualized* representations, including ELMo (Peters et al., 2018), GPT (Radford et al., 2018), BERT (Devlin

et al., 2019), and many others. The underlying idea is that, instead of learning a fixed vector to represent each word, we learn a function (usually some variant of a language model) that takes a word and its surrounding context and calculates a context-dependent vector as the meaning representation for the word. While relying heavily on the scale of the language model (up to billions of parameters) and the scale of the pre-training data (up to hundreds of Gigabytes of raw text), these contextualized representations have achieved great success on many tasks (e.g., semantic role labeling, textual entailment, reading comprehension, summarization, commonsense reasoning), without the need of complex neural architectures on top of them.

Although neural models powered by pre-trained embeddings are approaching human performance on many natural language understanding tasks, like the ones in the GLUE (Wang et al., 2019b) and SuperGLUE (Wang et al., 2019a) benchmarks, they do not necessarily understand the semantics of language. Jia and Liang (2017) show that when an adversarial sentence irrelevant to the question is appended to the passage in the SQuAD (Rajpurkar et al., 2016) reading comprehension dataset, the average F_1 score of sixteen published models drops significantly from 75% to 36%. Wallace et al. (2019) show that a lot of models can be deceived by a universal adversarial trigger. For example, a trigger consisting of some certain subword tokens will lead the pre-trained GPT-2 model (Radford et al., 2018) to always generate racist output even when conditioned on non-racial contexts. Such findings suggest that these models are probably only doing shallow pattern matching instead of semantic reasoning on most of the tasks, and the success might be partly due to easily exploitable biases in the datasets.

We argue that one important missing piece in more robust and generalizable models is to learn latent representations from semantic knowledge. Namely, in addition to training the model in an end-to-end fashion with the objective of some downstream tasks, we should strengthen the model with linguistic knowledge of semantic structure, for example, event structures, predicate-argument relations, or entity coreference. Our hypothesis is that such semantic knowledge will introduce beneficial inductive biases into the models and discourage them from exploiting surface-level cues in the datasets.

We start by focusing on one certain type of semantic knowledge, the implicit

predicate-argument relation, as it is a widespread phenomena in natural language while being understudied in recent research, partly due to the scarcity of training data (Gerber and Chai, 2010). We propose a cloze task to create synthetic training data for the problem, and study two different approaches of learning latent representations to infer implicit arguments from raw text. Both these approaches benefit from the structural knowledge of events and entities, and we also find that an auxiliary supervision derived from event structures on a multi-hop attention mechanism greatly boost performance on the more challenging cases where an event predicate has more than one implicit argument.

Then we shift our focus to models that take linguistic knowledge into account in a more latent manner. We show that by applying semantic structure as supervision for self-attention computation, a relatively small model can achieve significantly better results than the largest pre-trained GPT-2 model (Radford et al., 2018) on the LAMBADA task (Paperno et al., 2016), a language modeling task that is specifically designed to require the knowledge of broader discourse context in order to make the correct prediction even for human subjects. We further extend the idea to large-scale pre-trained language models, and discuss some ongoing experiments on the MRQA 2019 Shared Task (Fisch et al., 2019) aimed at making BERT (Devlin et al., 2019) more generalizable by injecting semantic knowledge during fine-tuning.

1.1 Thesis Outline

The remainder of the thesis is organized as follows:

- In Chapter 2, we review the necessary background information to understand this thesis, starting with a discussion on several semantic representations that we are interested in investigating. We also briefly review some relevant neural network architectures. This chapter also discusses some related work on integrating knowledge into neural models.

- In Chapter 3, we introduce the task of inferring implicit arguments, and propose an argument cloze task to address the data sparsity issue in training neural models. We also propose the EVENTCOMP model that draws on narrative coherence and entity salience, and evaluate it on a naturally occurring

nominal implicit arguments dataset, as well as a larger synthetic evaluation dataset based on argument cloze.

- In Chapter 4, we discuss some limitations of the argument cloze task and modify the task accordingly. We propose a new model for implicit arguments, the Pointer Attentive Reader, by casting the problem as reading comprehension to model global coherence, and demonstrate improved performance on the same two evaluation datasets.

- In Chapter 5, we investigate whether semantic knowledge can be integrated into neural models in a more latent manner. We experiment with several types of semantic knowledge, and find that integrating the knowledge as supervision for self-attention benefits downstream tasks that require long-distance reasoning. On the LAMBADA dataset, we show that a relatively small model enhanced with coreference knowledge achieves the new state-of-the-art performance.

- In Chapter 6, we explore whether semantic knowledge can make large-scale pre-trained language models more generalizable to unseen domains. We propose three methods of integrating semantic knowledge, and experiment on the MRQA 2019 Shared task. We find that it is very hard to obtain significant and consistent improvement across different pre-trained language models.

- In Chapter 7, we summarize our contributions and findings, and propose some directions for further investigation.

1.2 List of Contributions

In the thesis, we make the following contributions.

Neural Models for Inferring Implicit Argument The problem of implicit arguments has been regarded as a very hard problem in the literature, and most prior work focuses on feature-based methods. We propose two neural models to infer implicit arguments from raw text, building on the structural representation of events and entities. The first model measures local narrative coherence between

pairs of events, while the second model captures global coherence between the target event with the implicit arguments and all context events. Both these models achieve good performance on the nominal implicit arguments dataset from Gerber and Chai (2010).

The Argument Cloze Task One of the major obstacles in predicting implicit arguments is the lack of training data, mainly due to the difficulty in human annotation. We propose an argument cloze task to address the data sparsity issue, by randomly removing an event argument that is part of a coreference chain, and then asking the model to recover the removed argument. The task allows us to automatically create large-scale training data from raw corpus data, after preprocessing the text with off-the-shelf dependency parsing and coreference resolution tools. We show that the synthetic training data, while being noisy, still enables us to train effective neural models to infer implicit arguments.

We also create a synthetic evaluation dataset from the OntoNotes corpus (Hovy et al., 2006), which contains gold annotations of dependency parses and coreference chains. This synthetic evaluation dataset is larger and more comprehensive than existing human-annotated datasets, and is used for case analysis and ablation studies to assess models' performance.

Semantic Knowledge as Supervised Self-Attention We propose to integrate semantic knowledge into neural models as supervision for self-attention. On the LAMBADA task designed to require long-distance reasoning (Paperno et al., 2016), we show that integrating the knowledge of coreference chains into a relatively small model via supervised attention outperforms a much larger pre-trained language model, demonstrating the efficacy of the proposed method.

Integrating Semantic Knowledge into Pre-trained Language Models We extend the idea of integrating semantic knowledge to large-scale pre-trained language models like BERT (Devlin et al., 2019). In addition to using semantic knowledge as supervision for self-attention, we explore two more alternatives of applying semantic knowledge: as edge probing instances in a multi-task learning objective, or as a semantically-informative masking strategy for language modeling. On

the MRQA 2019 Shared Task (Fisch et al., 2019) aimed at testing the generalization capabilities of QA systems, we observe some improvement on the out-of-domain performance of several BERT variants, though the improvement is inconsistent and insignificant. We also analyze some possible shortcomings of our current approach, and discuss some future directions.

Chapter 2

Background

This thesis concerns learning better representations from semantic knowledge. To give a background on related prior work critical to the topic, we begin with a discussion on several semantic representations that we are interested in as "knowledge". We then introduce some neural network architectures that we investigate. Finally, we review recent approaches of integrating knowledge into neural models.

This chapter is intended to give a general motivation and background useful for understanding the original contributions in this thesis. Discussions on the more detailed background and prior work will be presented in each subsequent chapter individually.

2.1 Semantic Knowledge

By semantic knowledge, we mean the structural meaning representations of words and sentences. Among the various types of meaning representations that have been studied in the field of computational semantics, we primarily focus on two of them: **semantic roles**, and **coreference resolution**. We also briefly discuss Abstract Meaning Representation (**AMR**) at the end.

2.1.1 Semantic Roles

Semantic roles model the relations between event predicates and their participating arguments. For example, in the sentence:

(2.1) Alice gives Bob a book.

The predicate *give* has three arguments: *Alice* as the GIVER, *Bob* as the RECIPIENT, and *a book* as the THINGGIVEN. This is considered a relaxed form of semantic parsing, as it only requires filling a closed set of semantic roles with textual spans, while semantic parsing converts the sentence in Example (2.1) into a fully symbolic expression (as in Neo-Davidsonian event semantics (Davidson, 1967, Parsons, 1990)):

$$\exists e, x.\ \textsc{Give}(e) \ \wedge\ \textsc{Giver}(e, \mathit{Alice}) \ \wedge\ \textsc{Recipient}(e, \mathit{Bob})$$
$$\wedge\ \textsc{ThingGiven}(e, x) \ \wedge\ \textsc{book}(x)$$

Early work attempts to model the shared semantic properties of typical role fillers across different predicates using a universal set of **thematic roles** (Fillmore, 1968), such as AGENT (the volitional causer of an event), THEME (the participant mostly directly affected by an event), and INSTRUMENT (an instrument used in an event). But it has proved difficult to define a standard set of roles that cover all possible predicate-argument relations. Therefore, more recent work has focused on alternative semantic role models that use role sets that are either verb-specific or frame-specific, including the **Proposition Bank** (**PropBank**, Palmer et al., 2005) and **FrameNet** (Baker et al., 1998).

PropBank Instead of trying to explicitly define thematic roles, Dowty (1991) proposes the idea of **proto-roles**, with the observation that there often exists distinct characteristics between the agent role and the patient role of an event. Therefore, the more an argument displays agent-like properties (volitional involvement in the event, causing an event of a change of state in another participant, being sentient or intentionally involved, etc.), the more likely the argument can be labeled as a **proto-agent**. Similarly, if an argument exhibits more patient-like properties (undergoing change of state, causally affected by another participant, stationary relative to other participants, etc.), it is more likely to be a **proto-patient**.

PropBank (Palmer et al., 2005) builds on this agent-patient distinction, and assigns a list of numbered roles to each verb, like ARG0, ARG1, ARG2, and so on. Generally, ARG0 is the *proto-agent*, and ARG1 is the *proto-patient*. The semantics of other roles are more specific to each sense of each verb. The definition of each role is given as an informal gloss in frame files. For example, the roles of GIVE.01 (the first sense of the verb *give*) are defined as:

- ARG0: giver
- ARG1: thing given

- ARG2: entity given to

Therefore, the sentence in Example (2.1) can be labeled as

(2.2) [ARG0 Alice] **gives** [ARG2 Bob] [ARG1 a book].

In addition to the numbered roles, PropBank also defines some non-numbered roles (ARGM-*) for modification or adjunct meanings, for example, ARGM-TMP (time), ARGM-LOC (location), ARGM-MNR (manner), and so on. These adjunct roles are universal across all verbs and can be used in combination with any numbered roles. For example:

(2.3) [ARG0 Alice] **studied** in [ARGM-LOC the library] [ARGM-TMP yesterday].

The PropBank project annotates all verbs in all sentences in the Penn TreeBank (PTB, Marcus et al., 1993). Another related project, NomBank (Meyers et al., 2004), adds annotations to noun predicates, for example:

(2.4) [ARG0 Alice]'s **gift** of [ARG1 a book] to [ARG2 Bob] ...

FrameNet The FrameNet project (Baker et al., 1998, Fillmore, 1968) is another attempt on addressing the shortcomings of defining a universal set of thematic roles. Unlike PropBank, where definitions of semantic roles are specific to each verb, FrameNet focuses on frame-specific roles. A **frame** is defined as of a set of predicates (**lexical units**) that share some common background knowledge, and a set of frame-specific semantic roles, called **frame elements** (including core elements and non-core elements, which roughly correspond to the numbered and adjunct roles in PropBank). Currently, there are about 1,000 frames and a corpus of more than 100,000 "exemplar sentences" annotated in FrameNet.

For example, the **GIVING** frame includes lexical units like[1]: `contribute.v`, `donate.v`, `donation.n`, `gift.n`, `give.v`, `hand over.v`, `pass out.v`, and so on. And the core frame elements are defined as:

- DONOR: The person that begins in possession of the THEME and causes it to be in the possession of the RECIPIENT.

[1] https://framenet.icsi.berkeley.edu/fndrupal/frameIndex

- RECIPIENT: The entity that ends up in possession of the THEME.
- THEME: The object that changes ownership.

Therefore, the following two sentences would have exactly the same annotations in FrameNet:

(2.5)
a. [DONOR Alice] **gives** [RECIPIENT Bob] [THEME a book].
b. [DONOR Alice] **hands** [THEME a book] **over** to [RECIPIENT Bob].

Semantic Role Labeling The task of Semantic Role Labeling (**SRL**), is to automatically identify the semantic roles of each argument of each predicate in a sentence. The standard evaluation for SRL computes precision, recall, and F-measure based on whether each word in the sentence is assigned the correct argument label. The two most widely used datasets for evaluation are **CoNLL-2005** (Carreras and Màrquez, 2005), based on the PropBank corpus (Palmer et al., 2005), and **CoNLL-2012** (Pradhan et al., 2013), based on the OntoNotes corpus (Hovy et al., 2006). Both these datasets adopt the PropBank style annotation scheme. Another evaluation dataset, CoNLL-2009 (Hajič et al., 2009), differs from the former two in that it only requires predicting the head word of an argument, rather than predicting the whole argument span.

Early work (Gildea and Jurafsky, 2000; 2002) relied on the constituency tree of the input sentence (either from an off-the-shelf parser or from human annotations on PropBank), and views semantic role labeling as a feature-based classification problem. To illustrate, the constituency tree will identify all predicates in the sentence. Then, for every possible combination of a predicate and a constituent, the model extracts a set of features to predict a semantic role label (or $\varnothing$ if the constituent is not an argument of the predicate. Typical features include: the lemma of the predicate verb and its part-of-speech (POS) tag, the type of the phrase (e.g., NP, PP), the head word of the constituent and its POS tag, the syntactic path from the constituent to the predicate, and so on. Pradhan et al. (2005) also try extracting syntactic features from the dependency parse of the sentence.

One major limitation of treating SRL as a classification problem is that sometimes the model output might violate some global constraints, for example, assigning the same label (e.g., ARG0) to multiple constituents, or assigning different

labels to overlapping constitutes. Therefore, some follow-up work performs constrained optimization as the last step to ensure global consistency. One common choice is to use integer linear programming (ILP) to find the solution that conforms best to the constraints (Punyakanok et al., 2008).

More recently, neural network approaches have been introduced to SRL, by casting the problem as a sequence labeling task using the BIO notation[2]. In algorithms like Zhou and Xu (2015), Marcheggiani et al. (2017), He et al. (2017; 2018), the input sentence is first encoded by multiple layers of bidirectional LSTM (Hochreiter and Schmidhuber, 1997); then a constrained decoding layer is applied on the LSTM output to produce valid BIO tags. Shi and Lin (2019) investigate the use of large-scale pre-trained language models, like BERT (Devlin et al., 2019), and achieve F_1 scores close to 90 on benchmark SRL datasets. See Section 2.2 for a more detailed discussion on these neural architectures.

2.1.2 Coreference Resolution

One of most prominent forms of ambiguity in natural language understanding is "referential ambiguity", that is, ambiguity in which mentions refer to which entities or events. An example is shown in Figure 2.1. The individual spans in brackets are called **mentions** or **referring expressions**. They can be pronouns (like *its*, *his*), proper nouns (like *Yasser Arafat's Palestinian Authority*, *Israel*), or nominals (like *the truce*, *his country*). Two or more mentions are said to be **coreferent**, if they refer to the same underlying entity, like *Israel* and *his country*. Each mention has a (possibly empty) set of **antecedents**, which are preceding mentions that are coreferent. The task of **coreference resolution**, is to group mentions that refer to the same underlying entity. The set of coreferent mentions is often called a **coreference chain** or a **cluster**. For example, in Figure 2.1, *the truce* and *the cease-fire* form a coreference chain.

Coreference resolution is considered a very challenging problem. First, different types of referring expressions require different types of reasoning: features most useful for resolving the antecedents of pronouns might not work as well on

[2]The BIO notation represents each token as either the beginning of an argument (B), the inside of an argument (I), or outside of all arguments (O). For example, the BIO tags for Example (2.2) would be: Alice(B-ARG0) gives(O) Bob(B-ARG2) a(B-ARG1) book(I-ARG1).

The clashes took place as [Yasser Arafat's Palestinian Authority]$_1$ reaffirmed [its]$_1$ commitment to [the truce]$_2$. [Israel]$_3$ is to withdraw [its]$_3$ forces from several areas over the next two days, and [an Israeli official]$_4$ says [[his]$_4$ country]$_3$ would honor [the cease-fire]$_2$ even if sporadic rock-throwing continues.

Cluster 1	Cluster 2	Cluster 3	Cluster 4
* Yasser Arafat's Palestinian Authority * its	* the truce * the cease-fire	* Israel * its * his country	* an Israeli official * his

Figure 2.1: An example of coreference resolution from the OntoNotes corpus (Hovy et al., 2006). Coreferring mentions are indicated by brackets with the same subscript. Note that mentions may be nested. Mention clusters (coreference chains) are color coded and shown at the bottom.

proper nouns and nominals. Also, linguistic knowledge from the text might not be enough, and sometimes we need world knowledge to disambiguate sophisticated coreferring mentions, as shown in the following example from the famous **Winograd Schema Challenge** (Winograd, 1972):

(2.6)
a. The city council denied the demonstrators a permit because **they** feared violence.
b. The city council denied the demonstrators a permit because **they** advocated violence.

In (a), the antecedent for the pronoun *they* is *the city council*, while in (b), it is *the demonstrators*. This requires understanding that city councils are perhaps more likely to fear violence while demonstrators might be more likely to advocate violence.

Task and Evaluation To formalize the task of coreference resolution: for each mention m_i from the text, assign it to a cluster z_i such that $z_i = z_j$ if m_i and m_j are coreferent, or assign it to $\varnothing$ if it refers to a "singleton" entity (an entity that is mentioned only once). The task can thus be viewed as a structured prediction problem with two stages: identifying all entity mentions from the text (**mention**

detection), and grouping the detected mentions into clusters (**mention clustering**). We will briefly discuss these two stages in following paragraphs.

The most popular benchmark dataset for coreference resolution is the **CoNLL-2012 Shared Task** (Pradhan et al., 2012), which is derived from the OntoNotes corpus (Hovy et al., 2006). To evaluate a coreference system, we compare a set of hypothesis clusters produced by the system against a set of gold clusters from human annotations. However, the comparison is not trivial, and there exists a wide variety of metrics for doing this comparison. Following Denis and Baldridge (2009), the CoNLL-2012 Shared Task evaluates participating systems using three of the most popular metrics: MUC (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), and Entity-based CEAF (Luo, 2005). The average F_1 score of these three metrics is also reported (commonly referred to as CoNLL F_1).

Mention Detection Some prior work on mention detection (Lee et al., 2013, Durrett and Klein, 2013) chooses a deterministic approach, by starting with all noun phrases and named entities and then applying some heuristics to prune them. This might be problematic, as false negative mentions cannot be recovered from the second stage of mention clustering. An alternative (Lee et al., 2017) is to consider all spans up to a certain length as mention candidates, and perform mention detection and clustering in a joint fashion.

Mention Clustering There have been several different classes of models for mention clustering. The **mention-pair** model (Soon et al., 2001) predicts a binary label $y_{i,j} \in \{0, 1\}$ for each pair of mentions (m_i, m_j) where $i < j$ on whether m_i is an antecedent of m_j. The main shortcoming of this model is that it makes classifications on each pair of mentions independently, without taking into account global information on other mentions and entities.

A subsequent approach, the **mention-rank** model (Denis and Baldridge, 2008, Rahman and Ng, 2009, Durrett and Klein, 2013, Lee et al., 2017), compares all antecedent candidates for each mention: For each mention m_i, the model performs a multi-class classification among $\{m_1, \ldots, m_{i-1}, \varnothing\}$ to determine the correct antecedent, where $m_1, \ldots, m_{i-1}$ are the list of preceding mentions, and $\varnothing$ indicates that m_i does not have any antecedent (thus is the first mention of a new cluster).

Another alternative is the **entity-based** model (Rahman and Ng, 2009, Clark and Manning, 2015; 2016): Instead of predicting whether two mentions are coreferent, a classifier is trained to decide whether a mention m_k is coreferent with a partial cluster c_j that precedes m_k, or whether two partial clusters c_i and c_j should be merged.

Neural Network Approaches More recently, neural network models have been adopted for coreference resolution, and show rapid progress on benchmark performance. Clark and Manning (2016) represent mentions and entities as distributional vectors instead of hand-crafted feature sets. Lee et al. (2017; 2018) propose an end-to-end approach combined with the ELMo embeddings (Peters et al., 2018), and for the first time achieve a CoNLL F1 score above 70. Joshi et al. (2019; 2020) further improve the system by integrating large-scale pre-trained language models like BERT (Devlin et al., 2019) and SpanBERT. Wu et al. (2020) achieve a CoNLL F1 score of 83.1 by formulating the problem as a span prediction task like in machine reading comprehension, which is, to the best of our knowledge, the current state-of-the-art performance on the CoNLL-2012 Shared Task.

2.1.3 Abstract Meaning Representation

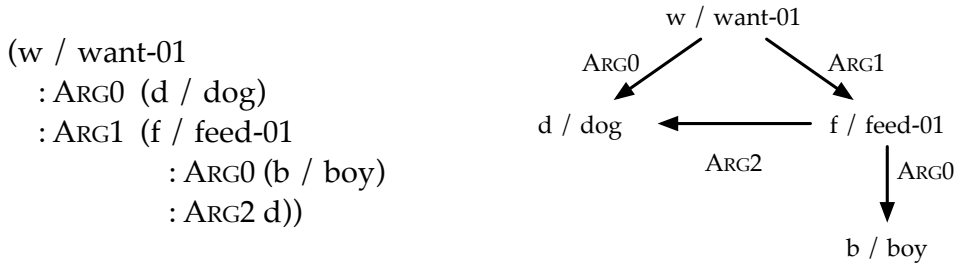
Traditional SRL annotation schemes, like PropBank and FrameNet, do not provide a unified view of all predicate-argument relations in a sentence. For example, consider the following sentence:

(2.7) The dog wants the boy to feed him.

PropBank would label this sentence as the following two propositions:

(2.8)
a. [$_{\text{ARG0}}$ The dog] **wants** [$_{\text{ARG1}}$ the boy to feed him].
b. The dog wants [$_{\text{ARG0}}$ the boy] to **feed** [$_{\text{ARG1}}$ him].

The **Abstract Meaning Representation (AMR)** (Banarescu et al., 2013) is an effort to merge these scattered annotations into a single graph structure for each sentence. Figure 2.2 shows an example of AMR in two notations: the PENMAN notation (Matthiessen and Bateman, 1991), and the graph notation.



(a) The PENMAN notation. (b) The graph notation.

Figure 2.2: An AMR annotation example in two notations.

In the AMR graph, each node is a variable, and each variable is an instance of a concept. This is represented by the slash notation, for example, *w / want-01* indicates that a variable *w* is an instance of the concept *want-01*. Variables can be reused, so that when we know that *him* refers to *the dog*, we can use the variable *d* to represent it in the graph. This also allows us to model intra-sentence coreference relations.

While AMR is a powerful meaning representation, it is also so comprehensive that it is very hard to train a parser to predict the graph. Even evaluating AMR parsing involves very complex algorithms (Cai and Knight, 2013). Early work on AMR parsing try graph-based models (Flanigan et al., 2014) or transition-based models (Wang et al., 2015) with hand-crafted features. More recently, Zhang et al. (2019a) introduce a neural parser for AMR by casting the problem as a sequence-to-graph transduction. However, these models generally do not work well on out-of-domain text, thus limiting its application to downstream tasks.

In this thesis, we do not directly use AMR graphs as semantic knowledge. In Chapter 5 and Chapter 6, we try combining the knowledge of predicate-argument relations and coreference relations, which can be loosely viewed as an approximation for AMR.

2.2 Neural Network Architectures

In this thesis, we experiment with several different types of neural networks. Here, we briefly review the basics of relevant neural architectures. More detailed

discussions on how these architectures are adapted to each problem are provided in subsequent chapters.

2.2.1 Sequence Modeling

Recurrent Neural Networks By nature, language is a sequence of variable length that unfolds in time, and traditional feedforward neural networks cannot handle such variable length input. Therefore, the **recurrent neural networks (RNNs)** are proposed to address this limitation, by introducing a recurrent connection on the hidden states from the previous time step to the current time step. The simplest form of RNN is called the **vanilla RNN** (or **Elman Networks**, Elman, 1990). To illustrate, we denote the inputs as $\{x_t\}_{t=1}^T$, the outputs as $\{y_t\}_{t=1}^T$, and the hidden states as $\{h_t\}_{t=1}^T$. At time step t, the Elman unit performs the following computation:

$$\begin{aligned} h_t &= f(U \cdot h_{t-1} + W \cdot x_t) \\ y_t &= g(V \cdot h_t) \end{aligned} \tag{2.1}$$

where f and g are non-linear transformation functions, U, W, V are trainable parameters. That is, at each time step, we update the hidden state based on the input state from the current time step and the hidden state from the previous time step.

In practice, two RNN architectures are often explored:

- **Stacked RNN**: We can stage multiple RNN layers together, using the hidden states of one layer as the input to a subsequent layer. To illustrate, assuming a stacked RNN with two layers, and the hidden states of each layer denoted as $\{h_t^1\}_{t=1}^T$ and $\{h_t^2\}_{t=1}^T$ respectively, then we have

$$\begin{aligned} h_t^1 &= f(U^1 \cdot h_{t-1}^1 + W^1 \cdot x_t) \\ h_t^2 &= f(U^2 \cdot h_{t-1}^2 + W^2 \cdot h_t^1) \\ y_t &= g(V \cdot h_t^2) \end{aligned} \tag{2.2}$$

- **Bidirectional RNN**: The vanilla RNN assumes that the hidden state at any time step depend only on the left context. In many application where the whole sequence is provided at once, information form the right context might

also be beneficial. Therefore, the bidirectional RNN can be viewed as the combination of a forward RNN and a backward RNN:

$$\begin{aligned}\overrightarrow{h_t} &= f(U^1 \cdot h_{\mathbf{t-1}} + W^1 \cdot x_t) \\ \overleftarrow{h_t} &= f(U^2 \cdot h_{\mathbf{t+1}} + W^2 \cdot x_t) \\ h_t &= \overrightarrow{h_t} \oplus \overleftarrow{h_t}\end{aligned} \tag{2.3}$$

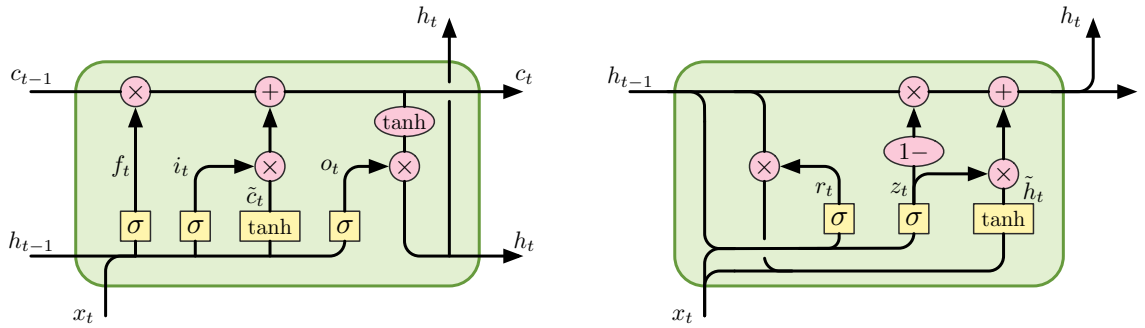
where $\oplus$ denotes vector concatenation.

RNNs can be trained via **back-propagation through time (BPTT)**, by unrolling all recurrent connections to form a big computation graph. However, on long sequence inputs, the gradients on early time steps are usually driven to zero, leading to the "vanishing gradients" problem. To overcome this issue, more complex variants have been designed with gating mechanisms, like the Long-short Term Memory networks (**LSTMs**, Hochreiter and Schmidhuber, 1997) and the Gated Recurrent Units (**GRUs**, Cho et al., 2014).

LSTMs In addition to the hidden states $\{h_t\}_{t=1}^T$, LSTMs add a set of memory cells $\{c_t\}_{t=1}^T$, and three gating functions: the *forget gate* f_t, the *input gate* i_t, and the *output gate* o_t. The update equations are:

$$\begin{aligned}
f_t &= \sigma(U_f \cdot h_{t-1} + W_f \cdot x_t) && \text{forget gate} \\
i_t &= \sigma(U_i \cdot h_{t-1} + W_i \cdot x_t) && \text{input gate} \\
\tilde{c}_t &= \tanh(U_c \cdot h_{t-1} + W_c \cdot x_t) && \text{update candidate} \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t && \text{memory cell update} \\
o_t &= \sigma(U_o \cdot h_{t-1} + W_o \cdot x_t) && \text{output gate} \\
h_t &= o_t \odot \tanh(C_t) && \text{output}
\end{aligned} \tag{2.4}$$

where $\odot$ is element-wise product. To explain, the *forget gate* controls what to carry from the previous memory cell; the *input gate* controls what to add from the current input; and finally, the *output gate* controls what to include in the hidden state. See Figure 2.3a for a graphical illustration.



(a) The LSTM Unit.

(b) The GRU Unit.

Figure 2.3: Illustration of LSTM and GRU. Reproduced from Christopher Olah's blog post: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

GRUs LSTMs introduce considerably more parameters compared to the vanilla RNN, leading to high memory footprint and time cost. To simplify, Cho et al. (2014) introduce the **Gated Recurrent Units**, by removing the need for memory cells and reducing the number of gates. The underlying idea is quite similar, with the *reset gate* controlling what to carry from the previous hidden state, and the *update gate* controlling what to add from the current input. A graphical illustration is shown in Figure 2.3b, with the equations as follows:

$$\begin{aligned} r_t &= \sigma(U_r \cdot h_{t-1} + W_r \cdot x_t) && \text{reset gate} \\ z_t &= \sigma(U_z \cdot h_{t-1} + W_z \cdot x_t) && \text{update gate} \\ \tilde{h}_t &= \tanh(U_\cdot(r_t \cdot h_{t-1}) + W \cdot x_t) && \text{output candidate} \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t && \text{output} \end{aligned} \tag{2.5}$$

2.2.2 Attention & Transformer

Seq2seq & Attention Recurrent neural networks have been widely used on natural language input, mostly in two ways:

- as a transducer (i.e., to produce one output for each input token), for example, in language modeling (Mikolov et al., 2010, Sundermeyer et al., 2012), or in sequence labeling tasks like part-of-speech tagging (Plank et al., 2016) and named entity recognition (Lample et al., 2016);

- or as a sequence encoder (i.e., to compress the whole input sequence into a fixed-size vector), for example, in machine translation (Sutskever et al., 2014), sentiment analysis (Dai and Le, 2015), and natural language inference (Bowman et al., 2015).

In the second approach, early methods simply use the last hidden state of RNN h_T as the representation for the whole sequence (or the concatenation of the last hidden states of the forward direction and the backward direction in bidirectional models: $\overrightarrow{h_T} \parallel \overleftarrow{h_1}$). However, a fixed-size vector does not scale up to represent an arbitrarily long sequence. Also, for problems like machine translation where the RNNs are typically employed in a **encoder-decoder** (also called **seq2seq**) architecture (Sutskever et al., 2014), that is, to first encode a variable-length input into a fixed-size vector, and then decode the vector back into a variable-length output (as shown in Figure 2.4a), this becomes more problematic as it is hard even for LSTMs to keep track of information over many time steps.

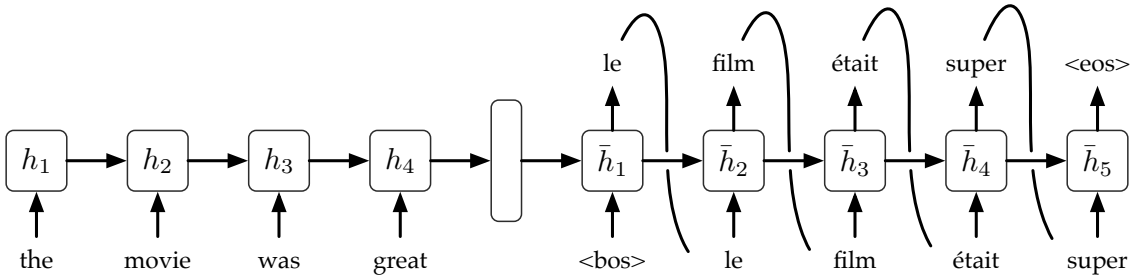
Bahdanau et al. (2015) propose the attention mechanism to allow the model to "look back" at any hidden state of the input sequence, as shown in Figure 2.4b. To illustrate, denote the encoder hidden states as $\{h_i\}$ and the decoder hidden states as $\{\bar{h}_j\}$. In the standard seq2seq model, at decoder time step j, we use the output from the previous time step y_{j-1} as input to update the RNN state:

$$\bar{h}_j = f(\bar{h}_{j-1}, y_{j-1}) \tag{2.6}$$

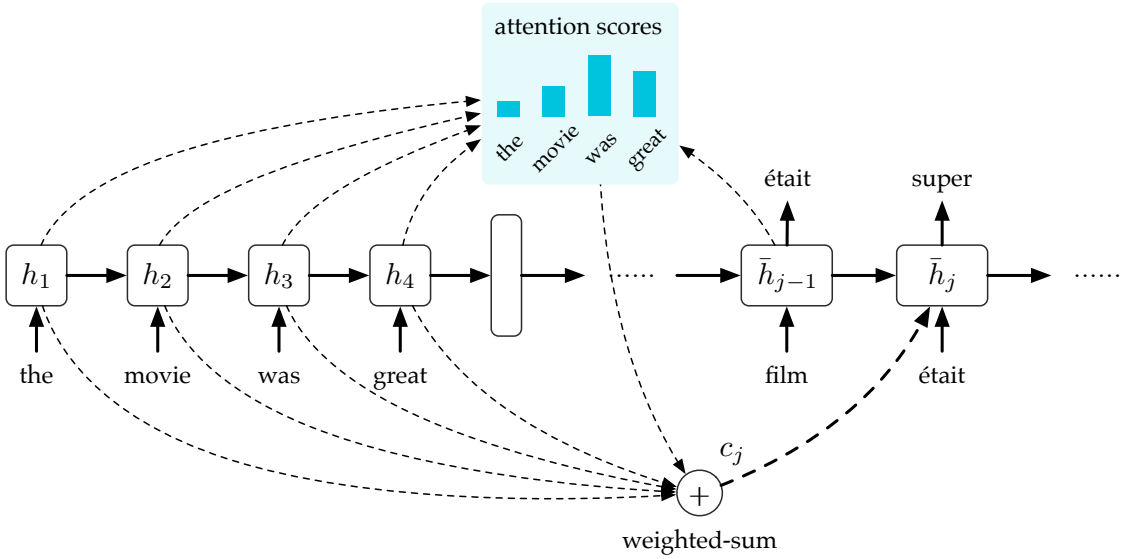
where the function f depends on the type of RNN being used (i.e., vanilla RNN, LSTM, or GRU), as discussed in Subsection 2.2.1. The attention mechanism adds an extra context vector c_j as input to function f:

$$\bar{h}_j = f(\bar{h}_{j-1}, y_{j-1}, c_j) \tag{2.7}$$

To get the context vector c_j, we first compute an attention score between the last decoder state $\bar{h}_{j-1}$ and every encoder state $\{h_i\}_{i=1}^{T}$, then use the normalized atten-



(a) The seq2seq model for machine translation from Sutskever et al. (2014).



(b) The seq2seq model with attention from Bahdanau et al. (2015).

Figure 2.4: The seq2seq model and the attention mechanism.

tion scores to compute a weighted-sum over all encoder states:

$$
\begin{aligned}
e_{ij} &= a(h_i, \bar{h}_j) \\
\alpha_{ij} &= \frac{\exp(e_{ij})}{\sum_{i'=1}^{T} \exp(e_{i'j})} \\
c_j &= \sum_{i=1}^{T} \alpha_{ij} h_i
\end{aligned}
\tag{2.8}
$$

where a is the attention function. Bahdanau et al. (2015) use the "additive" attention, and Luong et al. (2015) extend it with two more variants, "dot product" and

"bilinear":

$$
\begin{aligned}
a(h_i, \bar{h}_j) &= v_a^\top \tanh(W_a \cdot [h_i; \bar{h}_j]) && \text{additive} \\
a(h_i, \bar{h}_j) &= h_i^\top \cdot \bar{h}_j && \text{dot product} \\
a(h_i, \bar{h}_j) &= h_i^\top W_a \bar{h}_j && \text{bilinear}
\end{aligned} \tag{2.9}
$$

The **LSTM (or GRU) + Attention** combination has been successful on a number of tasks, like machine translation (Luong et al., 2015), reading comprehension (Hermann et al., 2015), summarization (Chopra et al., 2016), and semantic parsing (Jia and Liang, 2016).

Self-attention & Transformer While LSTM has become the most popular sequential encoder until very recently, one major shortcoming is that it cannot fully exploit the parallel computing power of modern GPUs due to the nature of its recurrent connections, making the training process very slow when the input sequences are long (e.g., several hundreds tokens). Vaswani et al. (2017) propose the **transformer** architecture with a stack of multi-head self-attention layers, and show strong performance on machine translation without any recurrent computation.

Unlike the traditional attention mechanism which attends from a decoder state to an encoder state, self-attention (Cheng et al., 2016) performs the attention computation from every encoder state to other encoder states. The motivation is quite intuitive: RNNs are designed to capture long-term dependencies, and attention is designed to further support this objective by allowing direct access to all hidden states. Therefore, by giving each token access to all other tokens at encoding time, the self-attention model should theoretically possess more expressing power than RNN-based encoders. The left column of Figure 2.5 illustrates the self-attention mechanism used in the transformer architecture.

At each self-attention layer, every token is projected into three vectors, the *query* Q, the *key* K, and the *value* V. An attention matrix is computed by the scaled dot-product between each *query* vector and each *key* vector, which is then used to compute a weighted-sum of the *value* vectors:

$$
\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V \tag{2.10}
$$

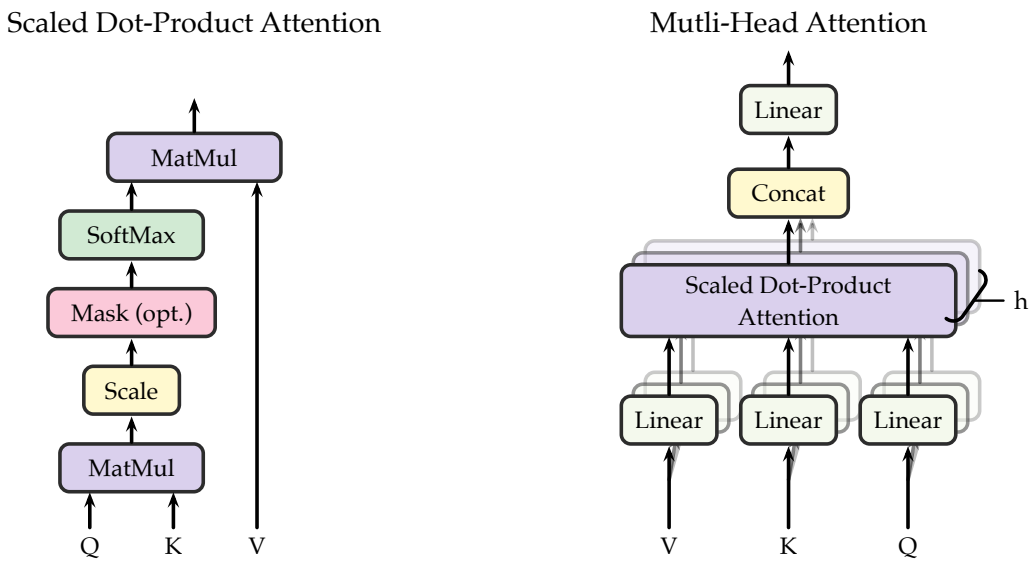


Figure 2.5: The multi-head self-attention used in the transformer architecture, from Vaswani et al. (2017).

where d_k is the dimension of query and key vectors. In the transformer architecture, each layer is a multi-head self-attention (the right column of Figure 2.5), by first splitting the Q, K, V vectors into small chunks and then concatenating the weighted-sum output of each chunk, analogous to the multiple convolutional filters used in CNNs.

The transformer architecture has been studied extensively in the past two years, and is the building block for many pre-trained contextualized models which will be discussed in the next subsection.

2.2.3 Pre-training & Contextualization

The first component of most neural network models for NLP is usually a mapping from discrete tokens to continuous vector representations (embeddings). Traditionally, pre-trained word embeddings like word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) have often been used as an initialization for the mapping. These embeddings are trained with large unlabeled corpora by learning word co-occurrence statistics, building on the distributional hypothesis that "a word is characterized by the company it keeps" (Firth, 1957). However, these fixed embeddings do not truthfully encode the meaning of words, because word

meanings are highly context-dependent.

Recently, we have seen a surge in the research of **contextualized embedding** models. McCann et al. (2017) propose **CoVe** (**Co**ntext **Ve**ctors), by pre-training a two-layer bidirectional LSTM on machine translation. The pre-trained LSTM is then used to compute context-dependent embeddings for input words, and show consistent improvement across many tasks. Peters et al. (2018) propose **ELMo** (**E**mbeddings from **L**anguage **Mo**dels), also by pre-training a two-layer bidirectional LSTM, but on the task of language modeling using the 1B Word Benchmark (Chelba et al., 2014), and achieve the best results at the time on a broad range of tasks, including reading comprehension, semantic role labeling, coreference resolution, and many others.

With the introduction of the transformer architecture (Vaswani et al., 2017), there has been a lot of interest in designing pre-trained transformers. Radford et al. (2018) propose **OpenAI GPT** (**G**enerative **P**re-**T**raining), by pre-training a 12-layer transformer as a generative language model, similar to ELMo, on the BooksCorpus (Zhu et al., 2015). A follow-up work, **GPT2** (Radford et al., 2019), use 10 times larger training corpora and more complex transformers (up to 1.5 billion parameters), and set new state-of-the-art on a number of language modeling tasks.

Probably the most popular pre-trained transformer model, **BERT** (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers, Devlin et al., 2019), enhances the GPT model by allowing bidirectional self-attention with a new "masked language modeling" (MLM) objective. To illustrate, given a chunk of text as input, BERT randomly replaces 15% of the tokens with a special *[MASK]* token (or another random token from the vocabulary), and then predicts the masked tokens in pre-training. BERT also adds another pre-training objective named "next sentence prediction" (NSP), by creating input in the form of *"[CLS]* $sent_a$ *[SEP]* $sent_b$ *[SEP]"*.In half of training samples, $sent_a$ and $sent_b$ are drawn from consecutive text; while the rest are drawn from random locations. The model is then asked to make a binary prediction on whether $sent_b$ does occur next to $sent_a$.

BERT is pre-trained on both the BooksCorpus (Zhu et al., 2015) and Wikipedia. The authors provides two model variants: $\text{BERT}_{\text{BASE}}$, with 12 self-attention layers, a hidden dimension of 768, and 12 attention heads per layer; $\text{BERT}_{\text{LARGE}}$, with 24 layers, 1024 hidden size, and 24 heads. Unlike ELMo where the LSTM weights

are kept frozen when fine-tuning on end tasks, fine-tuning BERT usually requires updating all parameters in the transformer (Peters et al., 2019b). Evaluation on the GLUE benchmark (Wang et al., 2019b) and some other datasets on reading comprehension (Rajpurkar et al., 2016; 2018) and commonsense reasoning (Zellers et al., 2018) show that BERT outperforms prior work by a huge margin.

There have been many follow-up models since then. **XLNet** (Yang et al., 2019b) propose a permutation language modeling objective to address the discrepancy between pre-training and fine-tuning in BERT. **RoBERTa** (Robustly optimized BERT, Liu et al., 2019) use 10 times more training data than BERT and dynamic masking on MLM pre-training. **SpanBERT** (Joshi et al., 2020) extend BERT with a "span boundary prediction" (SBO) objective by randomly masking contiguous spans of token. **T5** (**T**ext-**T**o-**T**ext **T**ransfer **T**ransformer, Raffel et al., 2019) cast every problem to a text-to-text format and utilize a huge pre-training dataset (750GB of text). **BART** (**B**idirectional and **A**uto-**R**egressive **T**ransformer, Lewis et al., 2019) present a encoder-decoder model pre-trained with multiple denoising autoencoder objectives.

2.3 Integrating Knowledge into Neural Models

In traditional feature-based machine learning approaches for NLP, some core annotation tasks are usually organized in a pipeline, where features and outputs from earlier tasks are usually used as input for later tasks. For example, in the Stanford CoreNLP pipeline (Manning et al., 2014), major component tasks are organized in the following order: part-of-speech tagging, named entity recognition, syntactic parsing, and coreference resolution. In this setup, linguistic knowledge is exploited inherently, for example, using syntactic structures to assist coreference resolution.

With the introduction of pre-trained embeddings and end-to-end neural models, research focus has shifted from "feature-engineering" (designing better features for each task) to "architecture-engineering" (tweaking network architectures to better capture task-specific features). The usage of linguistic knowledge has become less prevalent, especially in many downstream tasks, like question answering, sentiment analysis, information extraction, and so on. Nonetheless, there are

still some recent work on enhancing neural models with knowledge, which can be broadly divided into two categories according to the type of knowledge being utilized: **linguistic knowledge** or **background knowledge**.

2.3.1 Integrating Linguistic Knowledge

Syntactic Knowledge One major line of work in integrating linguistic knowledge to neural models is the use of syntactic knowledge for SRL. As discussed in Subsection 2.1.1, traditional feature-based systems usually rely on features from the syntactic structures like a constituency tree or a dependency tree (Gildea and Jurafsky, 2002, Pradhan et al., 2005). In the neural era, different approaches of incorporating syntax into neural models have also been explored.

One way is to encode syntactic structures into continuous vectors that will serve as additional input to SRL models, for example, by:

- Using LSTMs to encode the dependency paths between arguments and predicates (Roth and Lapata, 2016);
- Using graph convolutional networks (GCN, Kipf and Welling, 2017) to encode the syntactic tree structure (either dependency or constituency) of the input sentence (Marcheggiani and Titov, 2017; 2019);
- Using LSTMs to learn embeddings that represent the supertag label (Bangalore and Joshi, 1999) of each input token (Kasai et al., 2019).

Most of these approaches also require access to the syntactic structures at test time, either from gold annotations as provided by the evaluation tasks (Carreras and Màrquez, 2005, Pradhan et al., 2013), or from an off-the-shelf parser.

An alternative idea is to use syntactic structures as auxiliary training signals to guide the model to become more "syntax-aware".

- Strubell et al. (2018) propose the linguistically-informed self-attention (**LISA**) model, by training a multi-task transformer model for SRL together with several auxiliary objectives, including the prediction of POS tags, dependency heads, and dependency relations. The auxiliary objective for dependency head prediction is implemented as supervision for self-attention weights, by

encouraging a self-attention head to attend to each token's parent in a dependency tree.

- Swayamdipta et al. (2018) introduce **syntactic scaffolds** – auxiliary tasks designed to steer the model toward awareness of syntactic structure, and show that a syntactic scaffold to predict constituency labels can boost performance across PropBank SRL, FrameNet SRL, and coreference resolution.

- Cai and Lapata (2019) propose a joint learning approach to learn dependency information as an intermediate output in SRL prediction.

These models do not need syntactic structures at test time, though Strubell et al. (2018) also show that their model can further benefit from injecting a dependency tree during inference.

Semantic Knowledge There are also some very recent work on integrating semantic knowledge into neural models.

- Dhingra et al. (2018) propose to integrate coreference structure into standard GRUs via a "Coref-GRU" variant, by changing the hidden state update function to not only rely on the last time step (see Figure 2.3b), but also on the time step corresponding to the previous mention in the same coreference chain. Substituting standard GRUs with Coref-GRUs leads to performance gains on several reading comprehension tasks.

- Mihaylov and Frank (2019) enhance the QANet model (Yu et al., 2018) by integrating information from several semantic annotations, including SRL, coreference, and discourse relations. For each token, a set of annotation labels are extracted (e.g. SRL_ARG1, DISCREL_CAUSE), and the learned embeddings for these labels are used as additional input to the self-attention layers. On the NarrativeQA (Kočiský et al., 2018) reading comprehension dataset, knowledge from discourse relations provides the largest performance boost, while SRL and coreference show modest improvement.

- Similarly, Zhang et al. (2020) concatenate the BERT embeddings (Devlin et al., 2019) with the embeddings of semantic role labels, and show better results on the GLUE benchmark (Wang et al., 2019b) over the BERT baseline.

- Dai and Huang (2019) propose a regularization approach to integrate event knowledge and coreference relations into neural discourse parsing, by framing predicate-argument relations in SRL and coreferent mention relations as auxiliary loss in an end-to-end discourse parsing model, and show consistent improvement on almost all discourse relation classes.

2.3.2 Integrating Background Knowledge

Another type of widely used knowledge source in neural models is background knowledge. This can be either general knowledge bases about real-world entities, like **Wikipedia**, or structured knowledge graphs encoding factual or commonsense relations, such as:

- **WordNet** (Miller, 1995), a lexical database with expert annotations on a set of semantic relations (e.g., *hypernymy*, *hyponymy*, *antonymy*, etc) between concept nodes (called "synsets", by grouping words into sets of cognitive synonyms). For example, (*location*, *hypernym_of*, *city*).

- **ConceptNet** (Speer and Havasi, 2012, Speer et al., 2017), a large knowledge base for commonsense relations with over 8 million nodes and over 21 million edges. For example, (*ice*, *used_for*, *cooling*).

- **NELL** (Mitchell et al., 2015), an automatically constructed web knowledge base storing beliefs about entities. For example, (*New York*, *located_in*, *United States*).

One common strategy is the use of relational commonsense knowledge to facilitate natural language understanding, on tasks like reading comprehension (Yang and Mitchell, 2017, Mihaylov and Frank, 2018, Bauer et al., 2018) or natural language inference (Weissenborn et al., 2017, Chen et al., 2018). This usually follows a retrieve-then-encode paradigm, that is, to retrieve information from the knowledge graph and then encode it into the neural network. The form of the retrieved information can be either relevant concepts and facts (Yang and Mitchell, 2017, Mihaylov and Frank, 2018, Yang et al., 2019a), or parts of the graph (edges, logical paths, or sub-graphs) that connect words from the input text (Chen et al., 2018, Bauer et al., 2018, Wang and Jiang, 2019, Qiu et al., 2019, Lin et al., 2019). The

encoding step is performed by either an attention computation over the retrieved information, or by a graph updating step with GCNs (Kipf and Welling, 2017) or graph attention networks (Veličković et al., 2018) when sub-graphs are retrieved. Another alternative is to convert the retrieved information into free-form texts, which will be used as addition training input to refine word embeddings (Weissenborn et al., 2017).

There has also been prior work on integrating entity knowledge into neural models. Ahn et al. (2016), Yang et al. (2017), Ji et al. (2017), Logan et al. (2019) propose several variants of entity-aware language models, by adding external information about entities mentioned in the text into generative language modeling. More recently, with the introduction of pre-trained language models (see Subsection 2.2.3), some extensions of BERT have been studied to incorporate entity knowledge during pre-training, such as ERNIE (Zhang et al., 2019b), KnowBERT (Peters et al., 2019a), and Pretrained Encyclopedia (Xiong et al., 2020). We will discuss them in more detail in Section 6.2.

Chapter 3

Inferring Implicit Arguments by Local Coherence

This chapter introduces a method to infer implicit predicate-argument structures from raw text by exploiting local narrative coherence between pairs of events, and an argument cloze task to automatically create synthetic data for implicit arguments at scale. This is a first step to study how the semantic knowledge of events and entities can benefit the inference of implicit arguments. The work in this chapter has been published in Cheng and Erk (2018), where I developed the models, conducted the experiments and analysis, and wrote the paper, under the advice of Katrin Erk. All work in this chapter constitutes original contributions.

3.1 Chapter Overview

Predicate-argument tuples describe "who did what to whom", and are an important latent structure to extract from text. Such tuples can be useful in many tasks, for example Open Information Extraction (Etzioni et al., 2008) and Reading Comprehension (Hermann et al., 2015). This extraction is straightforward when arguments are syntactically connected to the predicate, but much harder in the case of *implicit arguments*, which are not syntactically connected to the predicate and may not even be in the same sentence. These cases are not rare; they can be found within the first few sentences on any arbitrary Wikipedia page (a common knowledge source for open information extraction), for example[1]:

(3.1) Twice in the late 1980s *Gillingham* came close to winning promotion to the second tier of English football, but a decline then set in ...

Here is another example from the reading comprehension dataset of Hermann et al. (2015):

[1] https://en.wikipedia.org/wiki/History_of_Gillingham_F.C.

(3.2)

Text: More than 2,600 people have been infected by *Ebola* in Liberia, Guinea, Sierra Leone and Nigeria since the outbreak began in December, according to the World Health Organization. Nearly 1,500 have died.

Question: The X outbreak has killed nearly 1,500.

In Example (3.1), *Gillingham* is an implicit argument to *decline*. In Example (3.2), it is *Ebola* that broke out, and *Ebola* was also the cause of nearly 1,500 people dying, but the text does not state this explicitly. *Ebola* is an implicit argument of both *outbreak* and *die*, which is crucial to answering the question. Generally, predicates with implicit arguments can be nouns, as *decline* and *outbreak*, or verbs, as *died*, in the above examples.

Implicit argument prediction as a machine learning task was first introduced by Gerber and Chai (2010) and Ruppenhofer et al. (2010), who each annotated a dataset for the task. However, neither of these two datasets contains more than 1,000 examples, making it a very hard task due to the limited size of training data. Most of the previous work on the task thus only utilized simple linear models while relied heavily on large numbers of hand-crafted features, making the models hard to generalize.

We hypothesize that narrative event coherence is crucial for implicit argument prediction. In Example (3.2), diseases are maybe the single most typical things to *break out*, and diseases also typically *kill* people, which is key to correctly answer the question. Built on this insight, we first address the data sparsity issue by introducing a simple argument cloze task (Section 3.3), which allows us to automatically generate synthetic training data at scale. We also create a large and comprehensive evaluation dataset following the same task setting, using the OntoNotes (Hovy et al., 2006) corpus. To take advantage of the large-scale training data, we then propose a neural model, **EVENTCOMP**, that draws on narrative coherence to predict implicit arguments. We also take into account entity salience features in our model, as the omitted arguments tend to be salient, as *Ebola* is in Example (3.2). Experiments on the Gerber and Chai (2010) dataset as well as the OntoNotes evaluation dataset show that our method achieves better performance than previous work and baselines.

3.2 Prior Work

3.2.1 Implicit Arguments

In this section, we review some prior work on implicit arguments, including the two main datasets, and some major follow-up work.

The G&C Dataset Gerber and Chai (2010) constructed a dataset (**G&C**) by selecting 10 nominal predicates from the NomBank (Meyers et al., 2004) corpus and manually annotating implicit arguments for all occurrences of these predicates. The resulting dataset is quite small, consisting of less than 1000 examples. An example for the nominal predicate *sale* is shown below:

(3.3) The average interest rate rose to 8.3875% at [Citicorp]$_{iarg_0}$'s $50 million weekly auction of [91-day commercial paper]$_{iarg_1}$, or corporate IOUs, from 8.337% at last week's [sale]$_{pred}$.

In the example, *Citicorp* is an implicit arg_0 (SELLER) of *sale*, and *91-day commercial paper* is an implicit arg_1 (THING SOLD) of *sale*. In an analysis of their data they found implicit arguments to be very frequent, as their annotation added 65% more arguments to NomBank. They also trained a linear classifier for the task relying on many hand-crafted features, including gold features from FrameNet Baker et al. (1998), PropBank Palmer et al. (2005) and NomBank. Gerber and Chai (2012) added more features and performed cross validation on the dataset, leading to better results.

The SEMEVAL-2010 Dataset Ruppenhofer et al. (2010) also introduced a similar dataset (**SEMEVAL-2010**) by annotating fillers for "null-instantiated" (implicit) semantic roles in several chapters of Arthur Conan Doyle's fiction works. Here is an example:

(3.4) In a lengthy court case the defendant was tried for [murder]$_{iarg_2}$. In the end, he was [cleared]$_{pred}$.

In the example, arg_2 (CHARGES) is a null-instantiated role of *cleared*, which can be filled by *murder* in the previous sentence. The task contains 3 phases: A model

would need to 1) identify such null-instantiated (NI) semantic roles, 2) determine whether or not a filler for the role can be accessible from the context (definite NI vs. indefinite NI), and 3) find the correct antecedents as fillers for DNIs. The dataset is even smaller than **G&C**, with 245 and 259 resolved DNIs in training and test data respectively.

Follow-ups There has since been much follow-up work on these two datasets, mainly in two directions.

(a) Some worked on proposing new methods for the task.

- Silberer and Frank (2012) cast the task as an anaphora resolution task, and applied a feature-based entity-mention model for coreference resolution to the SEMEVAL-2010 dataset.
- Laparra and Rigau (2013a) viewed the G&C task as a heuristic ranking problem rather than a classification problem.
- Laparra and Rigau (2013b) looked into how features from anaphora and coreference resolution can be applied to implicit argument resolution in the SEMEVAL-2010 task.
- Gorinski et al. (2013) built four different models for NI resolution and used majority voting to get the final prediction.
- Chiarcos and Schenk (2015) proposed to use memory-based learning to identify null-instantiated roles.
- Do et al. (2017) viewed each predicate along with its arguments as a sequence of $[pred, arg_0, arg_1, \dots]$, and trained a recurrent neural network for the task with an objective similar to language modeling.

(b) Other people focused on alleviating the size limitation by proposing methods for creating additional training data.

- Silberer and Frank (2012) used datasets with manually annotated coreference (Hovy et al., 2006) as additional training data. However the size of their additional training data is still limited given the amount of existing human annotations on coreference. In this chapter, we generate

large amounts of training data using raw text with automatically produced (though noisy) coreference labels (Subsection 3.5.2).

- Roth and Frank (2015) identified new instances of implicit arguments by aligning monolingual comparable texts, however the size is still similar to that of G&C and SEMEVAL-2010.
- Feizabadi and Padó (2015) combined the G&C and SEMEVAL-2010 datasets, using one as out-of-domain training data for another.
- Schenk and Chiarcos (2016) utilized raw text with automatically labeled semantic roles, but only used it to learn prototypical fillers in an unsupervised fashion.

The task is considered a very hard problem. On G&C, Gerber and Chai (2010) achieved an F_1 score of 42.3, while Gerber and Chai (2012) further boosted it up to 50.3, but still far below the human performance of 87.6. Some follow-up work (Laparra and Rigau, 2013a, Do et al., 2017) outperformed Gerber and Chai (2010), but none of them could beat Gerber and Chai (2012) on this task. SEMEVAL-2010 is even harder, possibly due to the smaller data size and that the task is more complex comparing to G&C. The two original participating systems in the challenge only reached F_1 scores of 1.2 and 1.4. Follow-up work exploited various techniques in modeling and data acquisition, but none of them could exceed an F_1 score of 20. Therefore, in our work we focus on evaluating on the G&C dataset.

There have been some other recent efforts in adapting datasets originally designed for other relevant semantic tasks to the problem of implicit arguments (Stern and Dagan, 2014, O'Gorman et al., 2018). We leave the experiments to future work, and discuss them in more detail in Subsection 7.1.1.

3.2.2 Narrative Coherence

Since we intend to exploit narrative coherence in predicting implicit arguments, in this chapter, we review some recent work on the field of **statistical script learning**, which studies the modeling and inference of prototypical sequences of narrative events and participants.

Early work on script learning dates back to the 1970s, when Schank and Abelson (1977) conducted a detailed analysis of structured scripts for the understand-

ing of world knowledge, with a well-known example being the *"restaurant script"*, describing the prototypical act sequence of a customer going into a restaurant as *entering, ordering, eating, and exiting*. Most follow-up work in the 1980s and 1990s continued to use the notion of hand-annotated event structures, thus making it very hard to learn and inference such event sequences with a statistical model due to the lack of data.

Chambers and Jurafsky (2008; 2009) first introduced the idea of using the statistics of co-occurrences between events with coreferring arguments to infer a narrative chain of events. The events and chains can be obtained from raw text without manual annotation, in the following way:

1. Run a dependency parser on the input text, and then extract events by identifying all verbs and their arguments (e.g., *McCann* being the subject of *threw* in Figure 3.1a).

2. Run a coreference resolution engine on the input text, and then identify the entity that each event argument from step 1 refers to (e.g., all underlined mentions in Figure 3.1a refer to the same entity "McCann").

3. Group all events into narrative chains so that events in a single chain involve the same entity in their arguments (Figure 3.1b shows the narrative chain involving "McCann").

The advantage of such approach is that the narrative chains can be learned in an unsupervised manner (albeit after parsing and resolving coreference in the text), thus allowing large-scale statistical models to be trained and evaluated.

To evaluate systems' performance on inducing narrative events, Chambers and Jurafsky (2008) also proposed a new cloze task that requires narrative knowledge to solve, the **narrative cloze** task. Given a narrative chain consisting of a sequence of events sharing an argument (referred to as the protagonist), one event is removed at random from the chain, and the model is asked to predict the missing event in the context of all the remaining events. In the task, a narrative event is a tuple of the event predicate (mostly a verb) and the typed dependency of the protagonist: *(predicate, dependency)*.

An example is shown in Figure 3.1. Given a piece of raw text with **event verbs** marked bold and mentions of the protagonist marked with underline (Fig-

1. McCann **threw** two interceptions early.
2. Toledo **pulled** McCann aside and **told** him he'd **start**.
3. McCann quickly **completed** his first two passes.

(a) A piece of text about American football with McCann as the protagonist.

1. (threw, subject)
2. (pulled, object)
3. (told, object)
4. (start, subject)
5. (completed, subject)

(b) The extracted narrative chain.

1. (threw, subject)
2. (pulled, object)
3. ??
4. (start, subject)
5. (completed, subject)

(c) A narrative cloze example.

Figure 3.1: An example of the narrative cloze task.

ure 3.1a), a narrative chain with five events can be extracted as in Figure 3.1b. Then in the narrative cloze task, we could remove the third event *(told, object)* and ask the model to predict the missing event using the remaining four events (Figure 3.1c). In this chapter, we adopt a similar idea to define the **argument cloze** task for implicit argument prediction, discussed in Section 3.3.

Many follow-up papers on statistical script learning have evaluated on the narrative cloze task:

- Chambers and Jurafsky (2009) extended their original work by learning to induce the "*narrative schema*", that is, a set of narrative chains on different protagonists, in order to take into account all entities in a document when inferring missing events. However, their method still only represent events as bare verbs, rather than the interaction between verbs and arguments.

- Pichotta and Mooney (2014) further improved Chambers and Jurafsky (2009) to model an event as a tuple of a verb and multiple arguments: $v(e_s, e_o, e_p)$, where v is the verb, and e_s, e_o, and e_p are the subject, direct object, and prepositional object of v. For example, *Mary gave the book to John* would give an event structure as *give(Mary, book, John)*.

- Rudinger et al. (2015) showed that sequences of events can be efficiently

modeled by a log-bilinear language model.

- Pichotta and Mooney (2016a;b) proposed to use recurrent neural networks instead of simple co-occurrence statistics to infer missing events.

- Granroth-Wilding and Clark (2016) trained a compositional neural network to learn dense representations of events by composing the embeddings of verbs and arguments, which were then used to compute pairwise event coherence scores to infer missing events.

For our task we also aim to exploit narrative coherence: We want to predict how coherent a narrative would be with a particular entity candidate filling the implicit argument position. So we take the model of Granroth-Wilding and Clark (2016) as our starting point for the EVENTCOMP model. More details will be discussed in Section 3.4.

3.3 The Argument Cloze Task

As discussed in Section 3.1 and Subsection 3.2.1, the main bottleneck of the implicit argument prediction task is knowledge acquisition, because it requires a lot of human effort in annotating implicit arguments, and the existing human-annotated datasets are too small for the use of neural models. The most relevant previous effort on resolving the data bottleneck is by Silberer and Frank (2012), who exploited datasets with manually annotated coreference as additional training data, but the size of the data is still limited.

We here present the **Argument Cloze** task, taking inspiration from the narrative cloze task (Chambers and Jurafsky, 2008), which allows us to automatically generate large scale data for both training and evaluation of implicit argument prediction. In this task, given a list of events extracted from the text, we *randomly remove* an *entity* from an argument position of one *event*. The entity in question needs to appear in at least one other place in the text. The task is then for the model to pick, from all entities appearing in the text, the one that has been removed.

We first define what we mean by an *event*, then what we mean by an *entity*.

- Following common practice in recent script learning literature (Pichotta and Mooney, 2016a, Granroth-Wilding and Clark, 2016), we define an **event** e as

a tuple:

$$e = (v, s, o, p)$$

where v is the verbal predicate, s is the subject, o is the direct object, and p is the prepositional object along with the preposition. Here we only allow one prepositional argument in the tuple, to avoid variable length input in computing event representations[2]. Also, we represent each predicate / argument as a concatenation of the head word and the role, in order to distinguish different positions in the event tuple. For example, from *Mary gave the book to John* we would extract the event tuple as *(give-pred, Mary-subj, book-dobj, John-prep_to)*.

- By an **entity**, we mean a coreference chain with a length of at least two – that is, the entity needs to appear at least twice in the text. This is to ensure that the argument being removed appears in at least one other place in the text for the model to recover.

We extract event structures from raw text using dependency parses. Given the dependency labels defined in the Universal Dependencies[3], we extract the subjects, the direct objects, and the prepositional objects based on the following dependency labels respectively[4]:

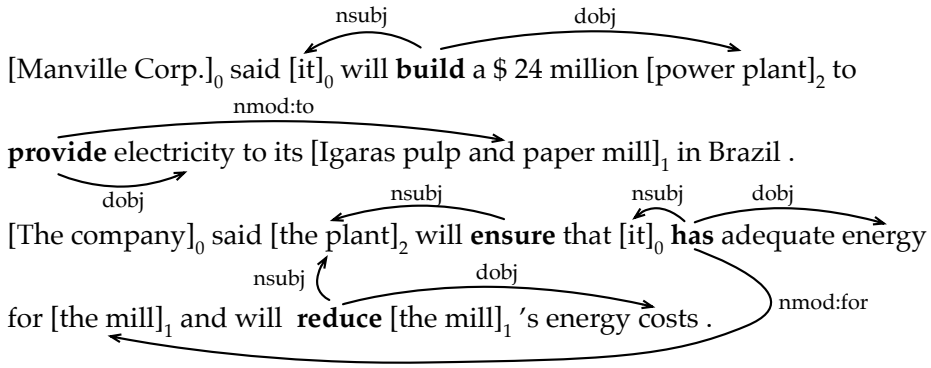
- Subjects: `nsubj` (nominal subjects), `nmod:agent` (agents of passive verbs), `nsubj:xsubj` (controlling subjects)
- Direct objects: `dobj` (direct objects), `nsubjpass` (passive nominal subjects)
- Prepositional objects: all `nmod:*` (nominal modifiers) except `nmod:agent`

For example, given a piece of raw text (Figure 3.2a), we first obtain the dependency relations (marked by arrows) and coreference chains (marked by subscript numbers) from the text, which are used to automatically extract a sequence

[2]In case of multiple prepositional objects, we select the one that is closest to the predicate. Also, we remove such constraint in our next work, as discussed in Chapter 4.

[3]http://universaldependencies.org.

[4]We still use the version 1 of UD as in http://universaldependencies.org/docsv1/, because version 2 was not available in the Stanford CoreNLP tool when this work was conducted. The core dependency labels are almost the same between v1 and v2 except a few name changes.



(a) A piece of raw text from the OntoNotes corpus (`english/nw/wsj_1278`), with dependency relations marked by arrows, coreference chains marked by subscript numbers, and predicates marked in bold.

e_0: (build-pred , x_0-subj , x_2-dobj , —)	x_0 = The company
e_1: (provide-pred , — , electricity-dobj , x_1-prep_to)	x_1 = mill
e_2: (ensure-pred , x_2-subj , — , —)	x_2 = power plant
e_3: (has-pred , x_0-subj , energy-dobj , x_1-prep_for)	
e_4: (reduce-pred , x_2-subj , cost-dobj , —)	

(b) Extracted events (e_0~e_4) and entities (x_0~x_2), using the above dependency and coreference annotations.

e_0: (build-pred , x_0-subj , x_2-dobj , —)	x_0 = The company
e_1: (provide-pred , — , electricity-dobj , ??-prep_to)	x_1 = mill
e_2: (ensure-pred , x_2-subj , — , —)	x_2 = power plant
e_3: (has-pred , x_0-subj , energy-dobj , x_1-prep_for)	
e_4: (reduce-pred , x_2-subj , cost-dobj , —)	

(c) An example instance of the argument cloze task for the *prep_to* argument of event e_1.

Figure 3.2: An example of the argument cloze task.

of events and a list of entities. In Figure 3.2b, e_0~e_4 are events, x_0~x_2 are entities. To construct a sample for the argument cloze task, we randomly remove an entity from an argument position. Therefore, arguments not in coreference chains, such as *electricity-dobj* in e_1 and *energy-dobj* in e_3, are not candidates for removal. An example of the argument cloze task is shown in Figure 3.2c. Here the *prep_to* argument of e_1 has been removed, and all entities appearing in the text are considered as candidates for the prediction.

The main differences between the original narrative cloze task and the argument cloze task are:

- We are not removing a whole event, but only an argument of an event. Also, due to the nature of implicit arguments (that they must be resolved to an existing entity in the context), we constrain the removal to the cases where the argument belongs to a coreference chain with a length of at least two, in order to make our cloze task more similar to the natural occurrences of implicit arguments.
- Unlike the original narrative cloze task in which the missing event is predicted from the whole vocabulary of events, we constrain our candidates to only entities appearing in the context, making the problem much more tractable.

Finally, although recent dependency parsers can produce quite accurate parse trees, even the state-of-the-art algorithm for coreference resolution is still very noisy. In the meanwhile, datasets with human-labeled gold coreference annotations are still very limited. Therefore, we use raw text with automatically generated dependency parses and coreference chains for creating training data (Subsection 3.5.2), and datasets with gold dependency and coreference annotations for creating additional evaluation data (Subsection 3.5.1).

3.4 Methods

3.4.1 Modeling Narrative Coherence

We model implicit argument prediction as selecting the entity that, when filled in as the implicit argument, makes the overall most coherent narrative. Suppose

we are trying to predict the direct object argument of some target event e_t. Then we complete e_t by putting an entity candidate into the direct object argument position, and check the coherence of the resulting event with the rest of the narrative. Say we have a sequence of events $e_1, e_2, \ldots, e_n$ in a narrative, and a list of entity candidates $x_1, x_2, \ldots, x_m$. Then for any candidate x_j, we first complete the target event to be

$$e_t(j) = (v_t, s_t, x_j, p_t), \quad j = 1, \ldots, m \tag{3.1}$$

where v_t, s_t, and p_t are the predicate, subject, and prepositional object of e_t respectively, and x_j is filled as the direct object. (Event completion for omitted subjects and prepositional objects is analogous.)

Then we compute the narrative coherence score S_j of the candidate x_j by: [5]

$$S_j = \max_{c=1,\ c \neq t}^{n} coh\left(\vec{e_t(j)}, \vec{e_c}\right), \quad j = 1, \ldots, m \tag{3.2}$$

where $\vec{e_t(j)}$ and $\vec{e_c}$ are representations for the completed target event $e_t(j)$ and one context event e_c, and coh is a function computing a coherence score between two events, both depending on the model being used. The candidate x_j with the highest score S_j is then selected as our prediction.

3.4.2 The EVENTCOMP Model

To model coherence (coh) between a context event and a target event, we build an event composition model consisting of three parts, as shown in Figure 3.3: event components are represented through **event-based word embeddings**, which encode event knowledge in word representations; the **argument composition network** combines the components to produce event representations; and the **pair composition network** compute a coherence score for two event representations.

This basic architecture is as in the model of Granroth-Wilding and Clark (2016). However our model is designed for a different task, argument cloze rather than narrative cloze, and for our task entity-specific information is more important. We therefore create the training data in a different way, as described in Subsec-

[5] We have also tried using the sum instead of the maximum, but it did not perform as well across different models and datasets.

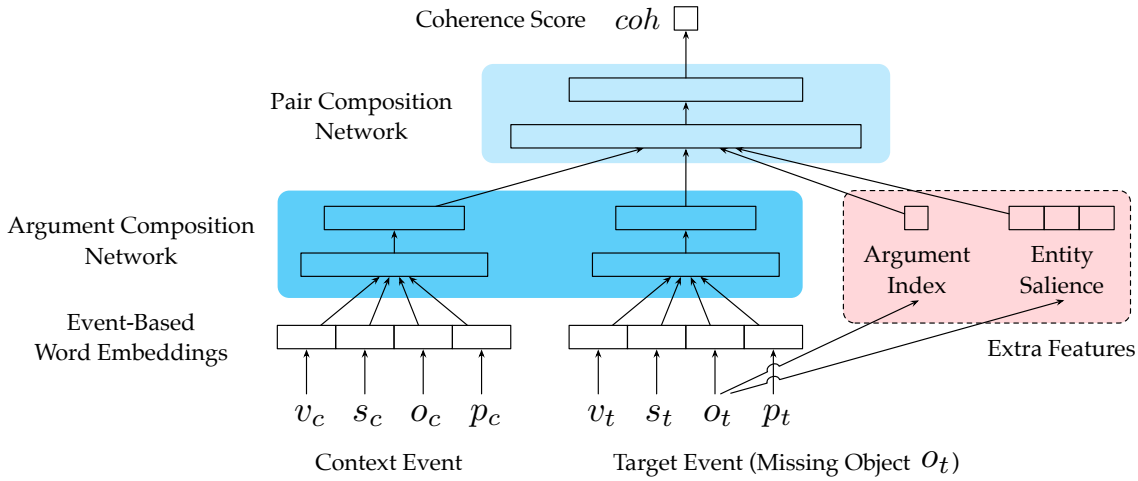


Figure 3.3: Diagram for the EVENTCOMP model. ***Input***: a context event and a target event. ***Event-Based Word Embeddings***: embeddings for components of both events that encodes event knowledge. ***Argument Composition Network***: produces an event representation from its components. ***Pair Composition Network***: computes a coherence score coh from two event representations. ***Extra Features***: argument index and entity salience features as additional input to the pair composition network.

tion 3.4.3. We now discuss the three parts of the model in more detail.

Event-Based Word Embeddings The model takes the word embeddings of both predicates and arguments as input to compute event representations. To better encode event knowledge at the word level, we train an SGNS (skip-gram with negative sampling) word2vec model (Mikolov et al., 2013) with event-specific information. For each extracted event sequence, we create a sequence with the predicates and arguments of all events in the sequence. An example of such a training sequence is given in Figure 3.4.

> … build-pred company-subj plant-dobj provide-pred electricity-dobj
> mill-prep_to ensure-pred plant-subj has-pred company-subj energy-dobj
> mill-prep_for reduce-pred plant-subj cost-dobj …

Figure 3.4: An example of event-based word2vec training sequence, constructed from the events and entities as shown in Figure 3.2b.

Argument Composition Network The argument composition network (marked in dark blue in Figure 3.3) is a two-layer feedforward neural network that composes an event representation from the embeddings of its components. All non-existent argument positions are filled with zeros.

Pair Composition Network The pair composition network (marked in light blue in Figure 3.3) computes a coherence score coh between 0 and 1, given the vector representations of a context event and a target event. The coherence score should be high when the target event contains the correct argument, and low otherwise. So we construct the training objective function to distinguish the correct argument from wrong ones, as described in Equation 3.3.

3.4.3 Training for Argument Prediction

To train the model to pick the correct candidate, we automatically construct training samples as event triples consisting of a context event e_c, a positive event e_p, and a negative event e_n. The context event and positive event are randomly sampled from an observed sequence of events, while the negative event is generated by replacing one argument of positive event by a random entity in the narrative, as shown in Figure 3.5.

Context: (build-pred, x_0-subj, x_2-dobj, —)	x_0 = The company
Positive: (reduce-pred, **x_2-subj**, cost-dobj, —)	x_1 = mill
Negative: (reduce-pred, **x_1-subj**, cost-dobj, —)	x_2 = power plant

Figure 3.5: A training sample as a triple of events, constructed from the events and entities as shown in Figure 3.2b.

We want the coherence score between e_c and e_p to be close to 1, while the score for e_c and e_n should be close to 0. Therefore, we train the model to minimize cross-entropy as follows:

$$\frac{1}{m}\sum_{i=1}^{m} -\log(coh(e_{ci}, e_{pi})) - \log(1 - coh(e_{ci}, e_{ni})) \tag{3.3}$$

where e_{ci}, e_{pi}, and e_{ni} are the context, positive, and negative events of the ith training sample respectively.

3.4.4 Entity Salience

Implicit arguments tend to be salient entities in the document. So we extend our model by entity salience features, building on recent work by Dunietz and Gillick (2014), who introduced a simple model with several surface level features for entity salience detection. Among the features they used, we discard those that require external resources, and only use the remaining three features, as illustrated in Table 3.1. Dunietz and Gillick found `mentions` to be the most powerful indicator for entity salience among all features. We expect similar results in our experiments, however we include all three features in our event composition model for now, and conduct an ablation test afterwards.

Feature	Description
`1st_loc`	Index of the sentence where the first mention of the entity appears
`head_count`	Number of times the head word of the entity appears
`mentions`	A vector containing the numbers of named, nominal, pronominal, and total mentions of the entity

Table 3.1: Entity salience features from Dunietz and Gillick (2014).

The entity salience features are directly passed into the pair composition network as additional input. We also add an extra feature for argument position index (encoding whether the missing argument is a subject, direct object, or prepositional object), as marked in red in Figure 3.3.

3.5 Experiments

3.5.1 Datasets

Previous hand-annotated implicit argument datasets (Gerber and Chai, 2010, Ruppenhofer et al., 2010) are very small. Therefore, in addition to evaluating on the G&C dataset, we also automatically create a large and comprehensive evaluation

dataset following the argument cloze task setup defined in Section 3.3. We first describe this synthetic evaluation dataset, then the G&C dataset.

Argument Cloze Evaluation Since the events and entities are extracted from dependency labels and coreference chains, we do not want to introduce systematic error into the evaluation from imperfect parsing and coreference algorithms. Therefore, we create the evaluation set from OntoNotes Hovy et al. (2006), which contains human-labeled dependency and coreference annotation for a large corpus. So the extracted events and entities in the evaluation set are gold. Note that this is only for evaluation; in training we do not rely on any gold annotations (we discuss our training data generation in Subsection 3.5.2).

There are four English sub-corpora in OntoNotes Release 5.0[6] that are annotated with dependency labels and coreference chains. Three of them, which are mainly from broadcast news, share similar statistics in document length, so we combine them into a single dataset and name it **ON-SHORT** as it consists mostly of short documents. The fourth sub-corpus is from the *Wall Street Journal* and has significantly longer documents. We call this sub-corpus **ON-LONG** and evaluate on it separately. Some statistics are shown in Table 3.2.

	ON-SHORT	ON-LONG
# doc	1027	597
# test cases	13018	18208
Avg # entities	12.06	36.95

Table 3.2: Statistics of the OntoNotes argument cloze datasets.

The G&C Dataset The G&C dataset (Gerber and Chai, 2010) consists of 966 instances of human-annotated implicit arguments on 10 nominal predicates.

To evaluate our model on G&C, we convert the annotations to the input format of our model as follows: We map nominal predicates to their verbal form, and semantic role labels to syntactic argument types based on the NomBank frame definitions, as defined in Table 3.3. The converted Example (3.3) (after mapping semantic role labels) is as follows:

[6]LDC Catalog No. LDC2013T19

(3.5) The average interest rate rose to 8.3875% at [Citicorp]$_{subj}$'s $50 million weekly auction of [91-day commercial paper]$_{dobj}$, or corporate IOUs, from 8.337% at last week's [sale]$_{pred}$.

For the nominal predicate *sale*, there are two arguments missing (*subj* and *dobj*). The model first needs to determine that each of those argument positions in fact has an implicit filler. Then, from a list of candidates (not shown here), it needs to select *Citicorp* as the implicit *subj* argument, and *91-day commercial paper* as the implicit *dobj* argument.

Nominal Predicate	Verbal Form	arg0	arg1	arg2	arg3	arg4
bid	bid	subj	prep_for	dobj	–	–
sale	sell	subj	dobj	prep_to	prep_for	prep
loan	loan	subj	dobj	prep_to	prep	prep_at
cost	cost	–	subj	dobj	prep_to	prep
plan	plan	subj	dobj	prep_for	prep_for	–
investor	invest	subj	dobj	prep_in	–	–
price	price	subj	dobj	prep_at	prep	–
loss	lose	subj	dobj	prep_to	prep_on	–
investment	invest	subj	dobj	prep_in	–	–
fund	fund	subj	dobj	prep	prep_on	–

Table 3.3: Mapping from the 10 nominal predicates in the G&C dataset to their verbal forms, and from the semantic role labels of each predicate to the corresponding dependency labels.

3.5.2 Implementation

We train our EVENTCOMP model using synthetic data as described in Section 3.3. For creating the training data, we do not use gold parses or gold coreference chains. We use the 20160901 dump of English Wikipedia[7], with 5,228,621 documents in total. For each document, we extract plain text and break it into paragraphs, while discarding all structured data like lists and tables[8]. We construct a sequence of events and entities from each paragraph, by running Stan-

[7] https://dumps.wikimedia.org/enwiki/

[8] We use the WikiExtractor tool at https://github.com/attardi/wikiextractor.

ford CoreNLP (Manning et al., 2014) to obtain dependency parses and coreference chains. We lemmatize all verbs and arguments. We incorporate negation and particles in verbs, and normalize passive constructions. We represent each argument by the corresponding entity's representative mention if it is linked to an entity, otherwise by its head lemma. We keep verbs and arguments with counts over 500, together with the 50 most frequent prepositions, leading to a vocabulary of 53,345 tokens; all other words are replaced with an out-of-vocabulary token. The most frequent verbs (with counts over 100,000) are down-sampled.

For training the event-based word embeddings, we create pseudo-sentences (Figure 3.4) from all events of all sequences (approximately 87 million events) as training samples. We train an SGNS word2vec model (Mikolov et al., 2013) with embedding size = 300, window size = 10, sub-sampling threshold = 10^{-4}, and negative samples = 10, using the Gensim package (Řehůřek and Sojka, 2010).

For training the event composition model, we follow the procedure described in Subsection 3.4.3, and extract approximately 40 million event triples as training samples[9]. We use a two-layer feedforward neural network with layer sizes 600 and 300 for the argument composition network, and another two-layer network with layer sizes 400 and 200 for the pair composition network. We use cross-entropy loss with ℓ_2 regularization of 0.01. We train the model using stochastic gradient descent (SGD) (Bottou, 2010) with a learning rate of 0.01 and a batch size of 100 for 20 epochs.

To study how the size of the training set affects performance, we downsample the 40 million training samples to another set of 8 million training samples. We refer to the resulting models as **EVENTCOMP-8M** and **EVENTCOMP-40M** respectively.

3.5.3 Results on Argument Cloze

For the synthetic argument cloze task, we compare our model with the following 3 baselines.

- **RANDOM** Randomly select one entity from the candidate list.

[9]We only sample one negative event for each pair of context and positive events for fast training, though more training samples are easily accessible.

- **MOSTFREQ** Always select the entity with highest number of mentions.
- **EVENTWORD2VEC** Use the event-based word embeddings described in Subsection 3.4.2 for predicates and arguments. The representation of an event e is the sum of the embeddings of its components, i.e.,

$$\vec{e} = \vec{v} + \vec{s} + \vec{o} + \vec{p} \tag{3.4}$$

  where $\vec{v}, \vec{s}, \vec{o}, \vec{p}$ are the embeddings of verb, subject, direct object, and prepositional object, respectively. The coherence score of two events in this baseline model is their cosine similarity. Like in our main model, the coherence score of the candidate is then the maximum pairwise coherence score, as described in Subsection 3.4.1.

The evaluation results are shown in Table 3.4. On ON-SHORT, the EVENTWORD2VEC baseline is much stronger than the other two, achieving an accuracy of 38.40%. In fact, EVENTCOMP-8M by itself does not do better than EVENTWORD2VEC, but adding entity salience greatly boosts performance. Using more training data (EVENTCOMP-40M) helps by a substantial margin both with and without entity salience features. Note that the entity salience features reported in Table 3.4 and Table 3.5 are computed from gold coreference chains, because the candidate entities in this task are also generated from gold chains (see Subsection 3.5.1). We acknowledge that this is not a realistic setting in the natural implicit argument prediction task. We were planning to conduct additional experiments with salience features computed from predicted coreference chains, but instead we moved to a different model architecture, detailed in Chapter 4, that implicitly takes (non-oracle) salience into account into account by its design. That latter architecture supersedes the experiments planned here. In the G&C evaluation of naturally occurring implicit arguments (Subsection 3.5.4), we use non-oracle entity salience features from the coreference chains predicted by the Stanford CoreNLP toolkit, which also bring about significant performance boost.

The ON-LONG dataset consists of OntoNotes data with much longer documents than found in ON-SHORT (Table 3.2). Although the overall numbers are lower than those for ON-SHORT, we are selecting from 36.95 candidates on average, more than 3 times more than for ON-SHORT. Considering that the accuracy

Accuracy (%)	ON-SHORT	ON-LONG
RANDOM	8.29	2.71
MOSTFREQ	22.76	17.23
EVENTWORD2VEC	38.40	21.49
EVENTCOMP-8M	38.26	18.79
+ entity salience	45.05	26.23
EVENTCOMP-40M	41.89	21.79
+ entity salience	**47.75**	**27.87**

Table 3.4: Evaluation results on the OntoNotes datasets.

of randomly selecting an entity is as low as 2.71%, the performance of our best performing model, with an accuracy of 27.87%, is quite good.

Ablation Analysis To see which of the entity salience features are important, we conduct an ablation test with the EVENTCOMP-8M model on ON-SHORT. From the results in Table 3.5, we can see that in our task, as in Dunietz and Gillick (2014), the entity mentions features, i.e., the numbers of named, nominal, pronominal, and total mentions of the entity, are most helpful. In fact, the other two features even decrease performance slightly.

Features	Accuracy (%)
no entity salience feature	38.26
– `mentions`	39.02
– `head_count`	**45.71**
– `1st_loc`	**45.65**
all entity salience features	45.05

Table 3.5: Ablation test on entity salience features. (Using EVENTCOMP-8M on ON-SHORT.)

We take a closer look at several of the models in Figure 3.6. Figure 3.6a breaks down the results by the argument type of the removed argument. On subjects, the EVENTWORD2VEC baseline matches the performance of EVENTCOMP, but not on direct objects and prepositional objects. Subjects are semantically much less diverse than the other argument types, as they are very often animate. A similar pattern is apparent in Figure 3.6b, which has results by the part-of-speech tag of

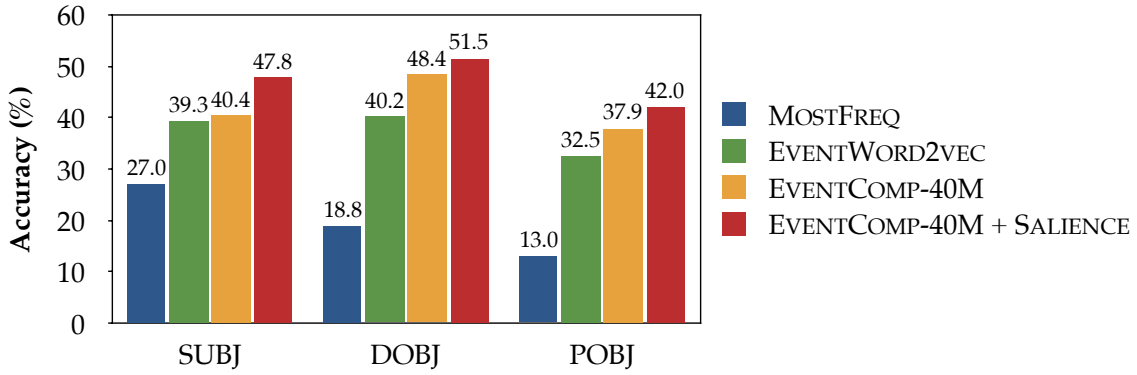
the head word of the removed entity. Note that an entity is a coreference chain, not a single mention; so when the head word is a pronoun, this is an entity which has only pronoun mentions. A pronoun entity provides little semantic content beyond, again, animacy. And again, EVENTWORD2VEC performs well on pronoun entities, but less so on entities described by a noun. It seems that EVENTWORD2VEC can pick up on a coarse-grained pattern such as animate/inanimate, but not on more fine-grained distinctions needed to select the right noun, or to select a fitting direct object or prepositional object. This matches the fact that EVENTWORD2VEC gets a less clear signal on the task, in two respects: It gets much less information than EVENTCOMP on the distinction between argument positions,[10] and it only looks at overall event similarity while EVENTCOMP is trained to detect narrative coherence. Entity salience contributes greatly across all argument types and parts of speech, but more strongly on subjects and pronouns. This is again because subjects, and pronouns, are semantically less distinct, so they can only be distinguished by relative salience.

Figure 3.6c analyzes results by the frequency of the removed entity, that is, by its number of mentions. The MOSTFREQ baseline, unsurprisingly, only does well when the removed entity is a highly frequent one. The EVENTCOMP model is much better than MOSTFREQ at picking out the right entity when it is a rare one, as it can look at the semantic content of the entity as well as its frequency. Entity salience boosts the performance of EVENTCOMP in particular for frequent entities.

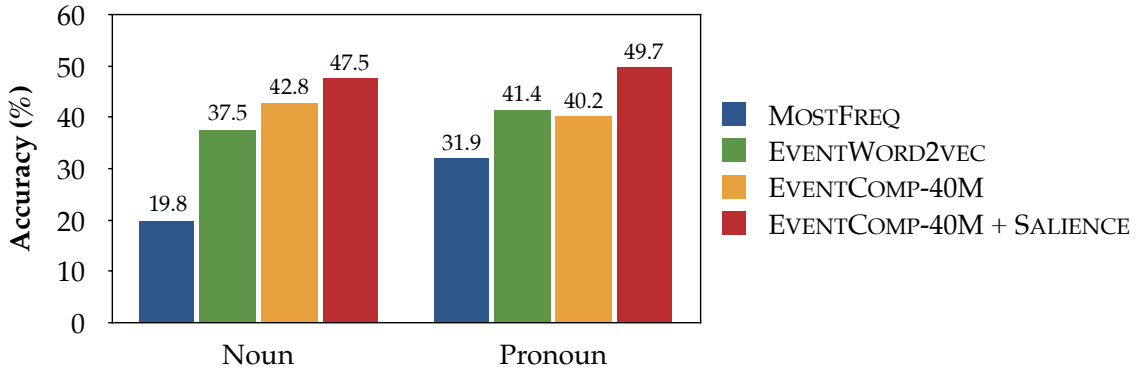
3.5.4 Results on G&C

The G&C data differs from the argument cloze data in two respects. First, not every argument position that seems to be open needs to be filled: The model must additionally make a **fill / no-fill decision**. Whether a particular argument position is typically filled is highly predicate-specific. Besides, the number of total missing argument positions (3737) is about 4 times larger than the number of gold implicit arguments (966), making the decision highly biased. As the small G&C dataset does not provide enough data to train our neural model on this task, we instead train a simple logistic classifier, the **fill / no-fill classifier**, with a small sub-

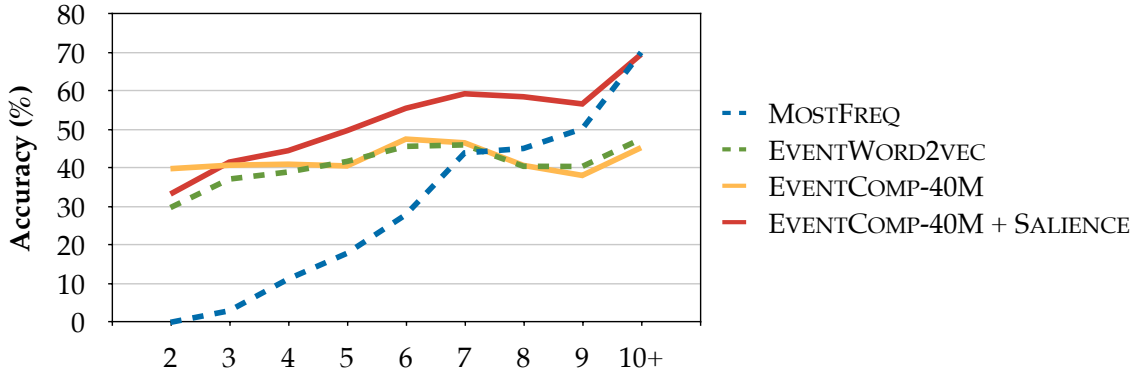
[10] As shown in Figure 3.4, the "words" for which embeddings are computed are role-lemma pairs.



(a) Accuracy by Argument Type



(b) Accuracy by POS of Head Word



(c) Accuracy by Entity Frequency

Figure 3.6: Performance of EventComp (with and without entity salience) and two baseline models by (a) argument type, (b) part-of-speech tag of the head word of the entity, and (c) entity frequency.

set of shallow lexical features used in Gerber and Chai (2012), to make the decision. These features describe the syntactic context of the predicate. We use only 14 features as shown in Table 3.6; the original Gerber and Chai (2012) model had more than 80 features, and our re-implementation, described below, has around 60.

#	Description
1	p itself.
2	p & p's morphological suffix.
3	p & $iarg_n$.
4	Verbal form of p & $iarg_n$.
5	Frequency of p within the document.
6	p & the stemmed content words in a one-word window around p.
7	p & the stemmed content words in a two-word window around p.
8	p & the stemmed content words in a three-word window around p.
9	p & whether p is before a passive verb.
10	p & the head of the following prepositional phrase's object.
11	p & the syntactic parse tree path from p to the nearest passive verb.
12	p & the part-of-speech of p's parent's head word.
13	p & the last word of p's right sibling.
14	Whether or not p's left sibling is a quantifier (many, most, all, etc.).

Table 3.6: Features used in the fill / no-fill classifier. This is a subset of features used by Gerber and Chai (2012). Here, p is the nominal predicate, $iarg_n$ is the integer n of the semantic role label of the implicit argument, as shown in Table 3.3, and the & symbol denotes concatenation.

The second difference is that in G&C, an event may have multiple open argument positions. In that case, the task is not just to select a candidate entity, but also to determine which of the open argument positions it should fill. So the model must do **multi implicit argument prediction**. We can flexibly adapt our method for training data generation to this case. In particular, we create extra negative training events, in which an argument of the positive event has been moved to another argument position in the same event, as shown in Figure 3.7. We can then simply train our EVENTCOMP model on this extended training data. We refer to the extra training process as **multi-arg training**.

As discussed in Subsection 3.2.1, none of the follow-up work on G&C could outperform the original results in Gerber and Chai (2012). Therefore, we only compare our models to that of Gerber and Chai (2012). However, their origi-

Context: (build-pred, x_0-subj, x_2-dobj, —)	x_0 = The company
Positive: (reduce-pred, **x_2-subj**, cost-dobj, —)	x_1 = mill
Negative: (reduce-pred, —, cost-dobj, **x_2-prep**)	x_2 = power plant

Figure 3.7: Event triples for training multi implicit argument prediction.

nal logistic regression model used many features based on gold annotation from FrameNet, PropBank and NomBank. To create a more realistic evaluation setup, we re-implement a variant of their original model by removing gold features, and name it GCAUTO. Results from GCAUTO are directly comparable to our models, as both are trained on automatically generated features. [11]

We present the evaluation results in Table 3.7. The original EVENTCOMP models do not perform well, which is as expected since the model is not designed to do the *fill / no-fill decision* and *multi implicit argument prediction* tasks as described above. With the fill / no-fill classifier, precision rises by around 13 points because this classifier prevents many false positives. With additional multi-arg training, F_1 score improves by another 22-23 points. At this point, our model achieves a performance comparable to the much more complex G&C re-implementation GCAUTO. Adding entity salience features [12] further boosts both precision and recall, showing that implicit arguments do tend to be filled by salient entities, as we had hypothesized. Again, more training data substantially benefits the task. Our best performing model, at 49.6 F_1, clearly outperforms GCAUTO, and is comparable with the original Gerber and Chai (2012) model trained with gold features.[13]

3.6 Chapter Summary

In this chapter, we address the task of implicit argument prediction. To support training at scale, we introduce a simple argument cloze task for which data

[11] To be fair, we also test adding the fill / no-fill classifier to GCAUTO. However the classifier only increases precision at the cost of reducing recall, and GCAUTO already has higher precision than recall. The resulting F_1 score is actually worse, and thus is not reported here.

[12] As discussed in Subsection 3.5.3, here we use non-oracle entity salience features from automatically predicted coreference chains.

[13] We also try fine-tune our model on the G&C dataset with cross validation, but the model severely overfit, possibly due to the very small size of the dataset.

	P	R	F_1
Gerber and Chai (2012)	57.9	44.5	50.3
GCAUTO	49.9	40.1	44.5
EVENTCOMP-8M	8.9	27.9	13.5
+ fill / no-fill classifier	22.0	22.3	22.1
+ multi-arg training	43.5	44.1	43.8
+ entity salience	45.7	46.4	**46.1**
EVENTCOMP-40M	9.4	30.3	14.3
+ fill / no-fill classifier	23.7	24.0	23.9
+ multi-arg training	46.7	47.3	47.0
+ entity salience	49.3	49.9	**49.6**

Table 3.7: Evaluating EVENTCOMP on G&C dataset.

can be generated automatically. We also introduce a neural model, EVENTCOMP, which frames implicit argument prediction as the task of selecting the textual entity that completes the event in a maximally narratively coherent way. The model prefers salient entities, where salience is mainly defined through the number of mentions. Evaluating on synthetic data from OntoNotes, we find that our model clearly outperforms even strong baselines, that salience is important throughout for performance, and that event knowledge is particularly useful for the (more verb-specific) object and prepositional object arguments. Evaluating on the naturally occurring data from Gerber and Chai (2010), we find that in a comparison without gold features, our model clearly outperforms the previous state-of-the-art model, where again salience information is important.

This chapter takes a first step towards predicting implicit arguments based on narrative coherence. We currently use a relatively simple model for local narrative coherence; in Chapter 4, we will turn to models that can test global coherence for an implicit argument candidate.

Chapter 4

Inferring Implicit Arguments by Global Coherence

This chapter introduces another method to infer implicit arguments, by modeling global narrative coherence between the target event and all preceding context events. The method is further enhanced by a multi-hop inference module to tackle cases with more than one implicit argument in a single event. This is a direct extension of the work in Chapter 3, providing another view on how to reason over event and entity structures to infer implicit arguments. The work in this chapter has been published in Cheng and Erk (2019), where I developed the models, conducted the experiments and analysis, and wrote the paper, under the advice of Katrin Erk. All work in this chapter constitutes original contributions.

4.1 Chapter Overview

In Chapter 3, we introduce a novel **argument cloze** task to address the data sparsity issue, and the **EVENTCOMP** model to predict implicit arguments, and achieve good performance on both the synthetic argument cloze evaluation data and the naturally occurring G&C dataset (Gerber and Chai, 2010). However, there are still some limitations of both the model and the task.

First, the model only learns local narrative coherence, in that it computes a pairwise coherence score between the target event and every context event separately to indicate how likely these two events exist in the same narrative chain, instead of jointly reasoning over all context events to measure global coherence.

Also, the model is not well designed to handle cases with more than one argument missing in a predicate-argument tuple. This is not rare in the natural occurrences of implicit arguments, as this is the case for more than 30% of the G&C dataset. In Example (4.1)[1], both arg_0 and arg_1 of the nominal predicate *sale* are implicit arguments. The EVENTCOMP model tries to infer these two implicit arguments **independently**, that is, to predict the filler for implicit arg_0 without any clue of implicit arg_1, and vice versa. However, plausible fillers for different implicit

[1]This is the same as Example (3.3), and we include it here just for easy reference.

arguments of the same event are typically mutually **interdependent**. In this example, knowing that *91-day commercial paper* is the implicit arg_1 of *sale* is an important clue to guess that *Citicorp* is the implicit arg_0, even for human readers.

(4.1) The average interest rate rose to 8.3875% at [Citicorp]$_{iarg_0}$'s $50 million weekly auction of [91-day commercial paper]$_{iarg_1}$, or corporate IOUs, from 8.337% at last week's [sale]$_{pred}$.

Further, many synthetic examples constructed from the argument cloze task do not closely resemble the implicit arguments in natural language, in that an argument would not likely to be left implicit unless a filler for the argument slot can be resolved from the preceding context. This is also evidenced by the two human-annotated implicit argument datasets (Gerber and Chai, 2010, Ruppenhofer et al., 2010), where only mention spans in the preceding context are considered as candidates of fillers. In the original argument cloze task, however, the correct fillers for a manually removed argument might only exist in the subsequent context.

Motivated by these observations, in this chapter, we view the task of implicit argument prediction as related to reading comprehension, and present another model, the **Pointer Attentive Reader (PAR)**, by combining a standard reading comprehension model (Hermann et al., 2015) with Pointer Networks (Vinyals et al., 2015).

- A predicate-argument tuple with the missing argument(s) is a query.
- All events in the preceding context consist the document.
- The answer to the query has to be located in the document.

To handle cases with multiple implicit arguments, we further extend the model with multi-hop inference, taking inspiration from multi-hop memory networks (Sukhbaatar et al., 2015), which allows the model to reason over the whole document multiple times to derive a more informative query.

In the following sections, we first review some prior work on reading comprehension and multi-hop reasoning (Section 4.2). Then we revisit the original argument cloze task and introduce some changes to the formulation of the task (Section 4.3), in order to address the statistical discrepancy between synthetic data

and natural data as discussed above. We present the model in Section 4.4 and experimental results in Section 4.5.

4.2 Prior Work

Hermann et al. (2015) first introduced neural models to reading comprehension tasks by collecting a large number of news articles paired with human-written bullet points, summarizing information contained in the article. To construct a corpus of document-question-answer triples, each bullet point is turned into a question by replacing one entity with a placeholder. They also proposed an Attentive Reader model, as shown in Figure 4.1. In the model, they first use bidirectional LSTMs (Hochreiter and Schmidhuber, 1997) to encode the document $(y(1), y(2), \dots)$ and the question (u) separately. Then, an attention mechanism $(s(1), s(2), \dots)$ is applied over the hidden representations of the document and query pair to derive a joint embedding (g), which is used to produce the final answer.

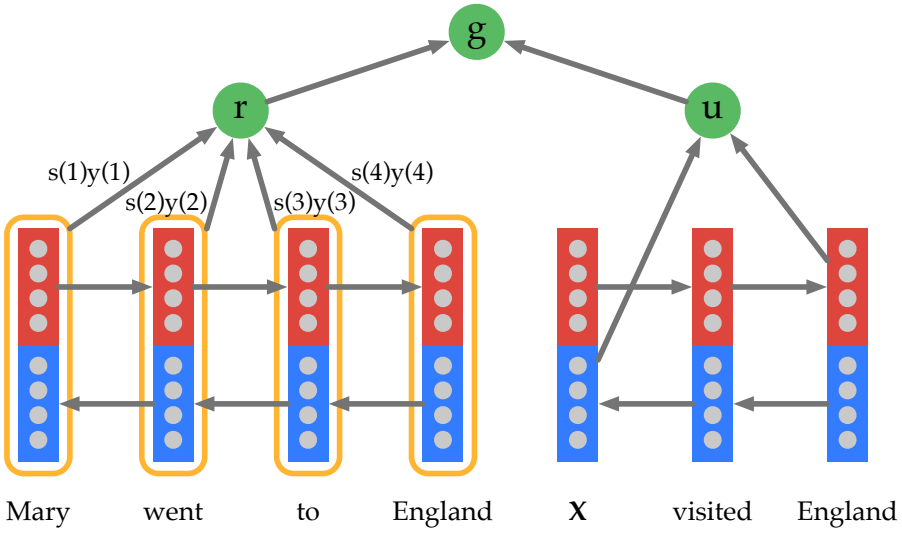


Figure 4.1: The Attentive Reader model from Hermann et al. (2015).

Since then there has been much follow-up work on constructing new datasets (Hill et al., 2016, Rajpurkar et al., 2016, Welbl et al., 2018, Yang et al., 2018) and proposing new models (Chen et al., 2016, Seo et al., 2017, Dhingra et al., 2017, Wang et al., 2017, Clark and Gardner, 2018) for reading comprehension. [2] In this chapter,

[2]More recently, large-scale pre-trained language models like BERT (Devlin et al., 2019) and XL-

we aim to take a first step in investigating how models in reading comprehension can be applied to better resolve implicit arguments than traditional models. We choose the Attentive Reader (Hermann et al., 2015) as our starting point, because it is simpler while still being a wide-used standard baseline.

Another related line of work in reading comprehension that is of particular interest to us is that on End-to-End Memory Networks (Sukhbaatar et al., 2015), which use multiple layers of attention computation (called "multiple hops") to allow for complex reasoning over the document input. An example of the bAbI task (Weston et al., 2015) in given in Figure 4.2. In each hop, the model updates the query representation with a weighted sum over all input sentences to gradually build a more informative query. For example, after the second hop, the model would know that the original query is equivalent to "What color is Brian?" after learning that "Brian is a frog" and "Greg is a frog". In our task, we want to use multi-hop inference to tackle cases where multiple implicit arguments are missing in one single event, as discussed in Subsection 4.4.3.

Story (16: basic induction)	**Support**	**Hop 1**	**Hop 2**	**Hop 3**
Brian is a frog.	yes	0.00	0.98	0.00
Lily is gray.		0.07	0.00	0.00
Brian is yellow.	yes	0.07	0.00	1.00
Julius is green.		0.06	0.00	0.00
Greg is a frog.	yes	0.76	0.02	0.00
What color is Greg? **Answer: yellow**		**Prediction: yellow**		

Figure 4.2: An example of multi-hop inference in Memory Networks from Sukhbaatar et al. (2015).

We also draw on Pointer Networks in that we view implicit argument prediction as a pointer to a previous mention of an entity. Vinyals et al. (2015) first proposed Pointer Networks as a variant of the conventional sequence-to-sequence models (Sutskever et al., 2014) that uses the attention distribution over input sequence directly as a "pointer" to suggest one preferred input state, instead of as a weight to combine all input states. This architecture has been applied to a number

Net (Yang et al., 2019b) have shown superior performance on a wide range of reading comprehension tasks, but they were not available at the time when this work (Cheng and Erk, 2019) was submitted (September 2018).

of tasks, including Question Answering (Xiong et al., 2017) and Machine Comprehension (Wang and Jiang, 2017).

4.3 Revisiting the Argument Cloze Task

The argument cloze task defined in Section 3.3 is a first step to address the data issue in training complex neural models for implicit argument prediction, and has shown good performance on both synthetic and natural evaluation datasets (Section 3.5). As a quick reminder, in the original setup, given a list of events extracted from the text, we construct an argument cloze example by randomly removing an argument of one event. The task is then to recover the removed argument, with all coreference chains (entities) as candidates. In this new model, we view the task as related to reading comprehension. Thus, we need to change the setup of the argument cloze task that would 1) better fit the nature of reading comprehension and 2) also make the cloze task more similar to naturally occurring implicit arguments, as discussed in Section 4.1.

Figure 4.3a shows the same set of events and entities as in Figure 3.2b, and we mark all arguments that may be removed in red boxes (essentially, such arguments belong to some coreference chain). However, for some of them (masked as gray in Figure 4.3b), the correct fillers only exist in subsequent events, because they are the first mentions of some coreference chain. Such argument should not be removed to construct synthetic argument cloze examples, as in natural language implicit arguments are most likely to occur when the fillers can be resolved from the preceding context, which is also the case in the two existing human-annotated implicit argument datasets (Gerber and Chai, 2010, Ruppenhofer et al., 2010). Therefore, to make the argument cloze task closer to the natural task, we only remove an argument if it co-refers with at least one argument in its preceding events, as marked in red boxes in Figure 4.3b.

Now since the ground truth filler for the removed argument must exist in the preceding context, to make it more realistic, the model should also only have access to the preceding context when making its prediction, similar to how human reason about implicit arguments in natural language. So we should consider all arguments in preceding events as candidates, instead of limiting to the ones that

e_0: (build-pred , x_0-subj, x_2-dobj, —)	x_0 = The company
e_1: (provide-pred , — , electricity-dobj , x_1-prep_to)	x_1 = mill
e_2: (ensure-pred , x_2-subj, — , —)	x_2 = power plant
e_3: (has-pred , x_0-subj, energy-dobj , x_1-prep_for)	
e_4: (reduce-pred , x_2-subj, cost-dobj , —)	

(a) The arguments to be removed in the original argument cloze task, where the events and entities are the same as in Figure 3.2b).

e_0: (build-pred , x_0-subj , x_2-dobj , —)	x_0 = The company
e_1: (provide-pred , — , electricity-dobj , x_1-prep_to)	x_1 = mill
e_2: (ensure-pred , x_2-subj, — , —)	x_2 = power plant
e_3: (has-pred , x_0-subj, energy-dobj , x_1-prep_for)	
e_4: (reduce-pred , x_2-subj, cost-dobj , —)	

(b) The arguments to be removed in the new setup.

Document ($e_0 \sim e_3$)

build-pred company-subj plant-dobj provide-pred electricity-dobj mill-prep_to ensure-pred plant-subj has-pred company-subj energy-dobj mill-prep_for

Query (e_4)

reduce-pred TARGET-subj cost-dobj

(c) An example of the modified argument cloze task, when viewed as reading comprehension in the form of a document-query pair.

Figure 4.3: Revisit the argument cloze example in Figure 3.2.

are part of some coreference chain, because without access to the subsequent context, the model would not know whether an argument actually co-refers with some other arguments or not. Therefore, we also change the evaluation of the task by considering candidates to be mentions, not coreference chains, and by considering only candidates that appear before the implicit argument, independent of their number of mentions.

We thus formalize the task as shown in Figure 4.3c. For an event (e_4) with a missing argument (*subj*), we concatenate the predicates and arguments of all preceding events (e_0~e_3) into a sequence and view it as the document, and we treat the target event with a special placeholder token (marked red) at the missing argument position as the query. The task is then to select any mention of the correct entity (marked blue) among the arguments appearing in the preceding document. A query may have multiple correct answers when there are multiple mentions of the removed entity, as shown in this example.

4.4 Methods

4.4.1 Pointer Attentive Reader

As discussed above, we view the task of implicit argument prediction as a variant of reading comprehension, in that we can treat the list of preceding events as a document and the target event with missing argument as a query. And we also draw on Pointer Networks and on multi-hop attention.

Most previous work on reading comprehension (Chen et al., 2016, Seo et al., 2017, Dhingra et al., 2017, Wang et al., 2017) can be viewed as extending the Attentive Reader model by Hermann et al. (2015). The Attentive Reader (Figure 4.1) first encodes the document and the query via separate recurrent neural networks to get a list of document word vectors and one query vector. The query vector is used to obtain an attention-weighted sum over all document word vectors, which is then combined with the query vector to make the final prediction.

In the case of implicit argument prediction, however, the task is to directly select one token (an argument mention) from the document input sequence as the filler for the missing argument. This suggests the use of Pointer Networks (Vinyals et al., 2015), a variant of the sequence-to-sequence model that uses the attention

distribution over input states to "point" to a preferred input state.

So we combine the ideas from Attentive Reader and Pointer Networks and propose the **Pointer Attentive Reader (PAR)** model for implicit argument prediction, as illustrated in Figure 4.4.

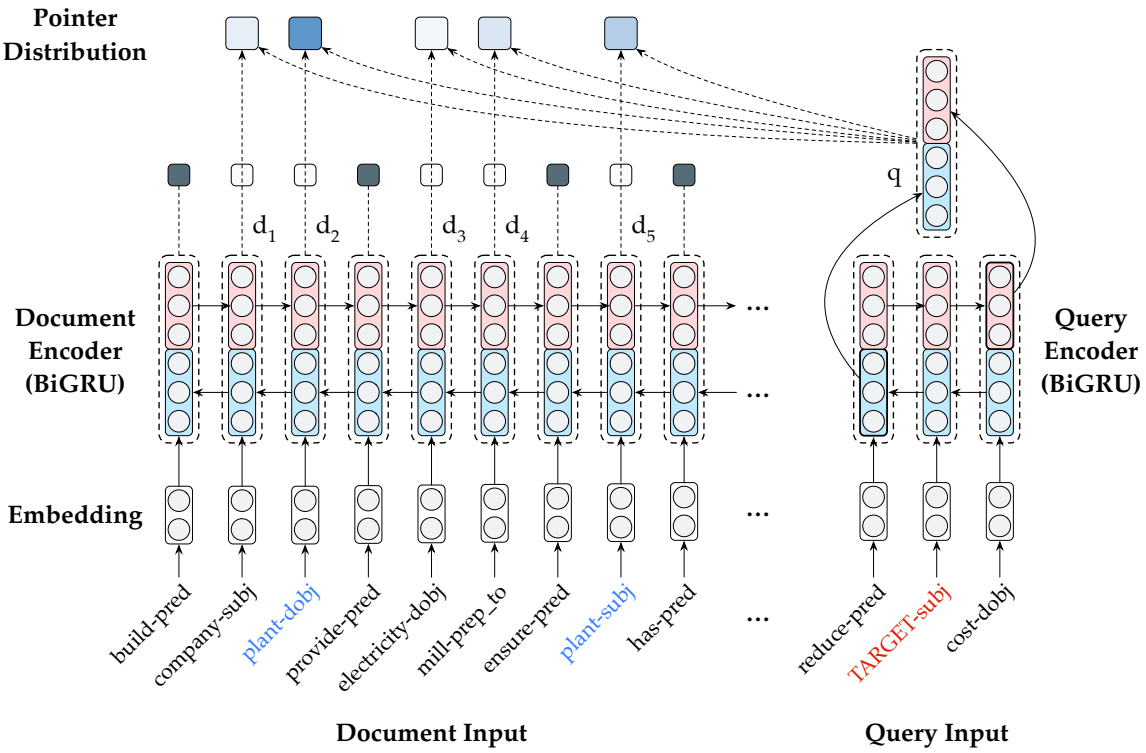


Figure 4.4: Pointer Attentive Reader (PAR). The **document encoder** produces a context-aware embedding for each argument mention via a BiGRU. The **query encoder**, similar to the document encoder, concatenate the last forward and backward hidden state to produce a single query vector. The attention scores between the query vector $\mathbf{q}$ and all argument mention embeddings $\mathbf{d_t}$ are normalized to a **pointer distribution** to select one filler for the missing argument in the query.

Embedding The document input and the query input, as discussed in Section 4.3, are both concatenated sequences of event components, represented as $[x_1^d, \ldots, x_{|D|}^d]$ and $[x_1^q, \ldots, x_{|Q|}^q]$ respectively (where $|D|$ and $|Q|$ are the numbers of tokens in document and query). The missing argument in the query is represented by a special placeholder token. Each token is then mapped to an embedding vector before being passed into the document encoder and query encoder.

Document Encoder The document encoder is a bidirectional single-layer Gated Recurrent Unit (BiGRU) (Cho et al., 2014). The forward and backward hidden state of each token are concatenated, with predicate tokens being masked out (as predicates are not considered as candidates), which gives us a list of context-aware embeddings of argument mentions: $[\mathbf{d_1}, \ldots, \mathbf{d_T}]$.

Query Encoder The query encoder is also a BiGRU similar to the document encoder, except that we concatenate the last forward hidden state and the last backward hidden state to get the single query vector $\mathbf{q}$.

Attention For each argument mention embedding $\mathbf{d_t}$, we compute an attention score a_t using the query vector $\mathbf{q}$ as[3]:

$$\begin{aligned} s_t &= \mathbf{v}^T \cdot \tanh(\mathbf{W}[\mathbf{d_t}, \mathbf{q}]) \\ a_t &= \text{softmax}(s_t) \end{aligned} \tag{4.1}$$

where $\mathbf{W}$ and $\mathbf{v}$ are learned parameters.

Finally, the attention scores $[a_1, \ldots, a_T]$ are directly used as pointer probabilities to select the most probable filler for the implicit argument.

4.4.2 Training Objective

Unlike conventional pointer networks where there exists a single target for the pointer, there could be multiple correct answers from the document input list in the implicit argument prediction task (as in the example in Figure 4.3c). Therefore, we train the model to maximize the "maximum correct" attention score. That is, with a list of attention scores $\mathbf{a} = [a_1, a_2, \ldots, a_T] \in \mathbb{R}^T$, and a binary answer mask $\mathbf{m_c} \in \mathbb{R}^T$ which has 1's for correct answer positions (e.g., *plant-dobj* and *plant-subj* in Figure 4.3c) and 0s elsewhere, we train the model with the following negative log likelihood (NLL) loss function:

$$L = -\log(\max(\mathbf{a} \circ \mathbf{m_c})) \tag{4.2}$$

[3]We have also tried bilinear attention and dot product attention (Luong et al., 2015), but got lower performance.

where $\circ$ is element-wise multiplication.

4.4.3 Multi-hop Attention

A single event can have more than one implicit argument, and in fact this is the case for over 30% of nominal predicates in the dataset of Gerber and Chai (2010). In such cases, we still treat one implicit argument as the target argument to be filled, and the other arguments are indicated to the model to be missing but not target, using a separate placeholder token. An example is shown in Figure 4.5, where target arguments are marked red, "missing but not target" arguments are marked bold, and answers to the target arguments are marked blue.

Document 1 (e_0 ~ e_2)
build-pred company-subj plant-dobj provide-pred electricity-dobj mill-prep_to ensure-pred plant-subj

Query 1 (e_3)
has-pred **MISS-subj** energy-dobj TARGET-prep_for

Document 2 (e_0 ~ e_2)
build-pred company-subj plant-dobj provide-pred electricity-dobj mill-prep_to ensure-pred plant-subj

Query 2 (e_3)
has-pred TARGET-subj energy-dobj **MISS-prep_for**

Figure 4.5: An example of document-query pairs for predicates with more than one implicit argument.

When there are multiple implicit arguments, this could make the query vector $\mathbf{q}$ lack enough information to compute the correct attention distribution, especially in the extreme case where only the predicate and placeholder tokens are present in the query input. To overcome this difficulty, we strengthen the model with the ability to reason over the document and query to infer the missing but non-target arguments and thus build a better query. We do this by extending the Pointer Attentive Reader model with multi-hop attention, inspired by the idea of end-to-end memory networks (Sukhbaatar et al., 2015). For example in Figure 4.5, we can

make the vector of Query 1 more informative by attending to all missing arguments of *has* in the first hop. We are not predicting the subject at this point, but could use it to help the final prediction of *TARGET-prep_for*. Figure 4.6 shows the 2-hop Pointer Attentive Reader model.

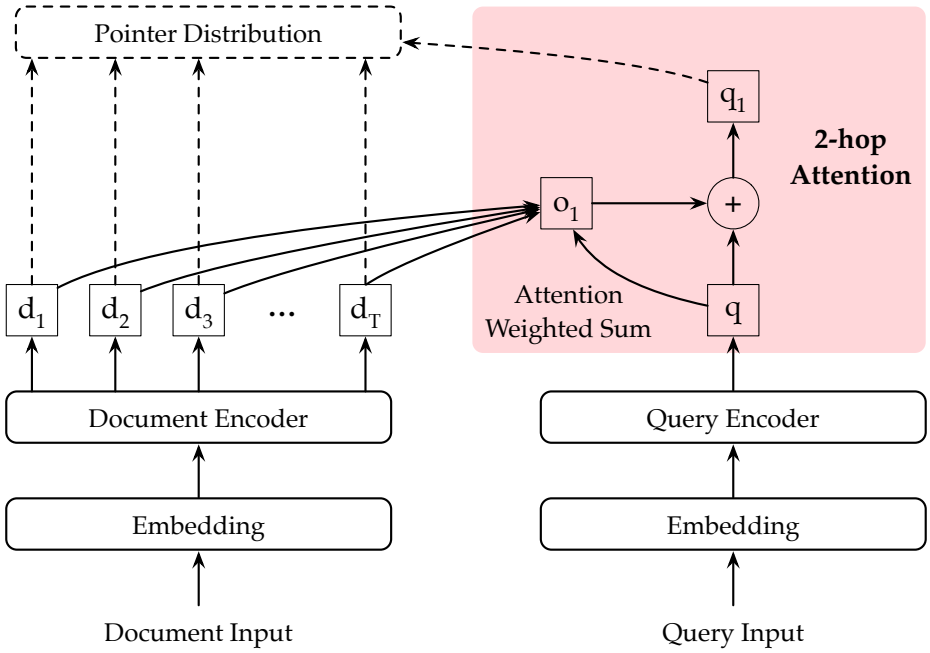


Figure 4.6: 2-hop Pointer Attentive Reader. The query vector $\mathbf{q}$ is first updated by an attention weighted sum $\mathbf{o_1}$ from all argument embeddings in the document, before used to compute the final attention distribution.

To make the query vector document-aware, we update the query vector $\mathbf{q}$, in each but the last hop, by an attention-weighted sum $\mathbf{o_1}$ over argument embeddings $[\mathbf{d_1}, \mathbf{d_2}, \ldots, \mathbf{d_T}]$:

$$\begin{aligned} s'_t &= \mathbf{v'}^T \cdot \tanh(\mathbf{W'}[\mathbf{d_t}, \mathbf{q}]) \\ a'_t &= \text{softmax}(s'_t) \\ \mathbf{o_1} &= \Sigma_{t=1}^{T} a'_t \cdot \mathbf{d_t} \\ \mathbf{q_1} &= \mathbf{o_1} + \mathbf{q} \end{aligned} \tag{4.3}$$

where $\mathbf{W'}$ and $\mathbf{v'}$ are learned parameters. Then in Equation 4.1 we use $\mathbf{q_1}$ instead of $\mathbf{q}$ to compute the final attention scores.

In this chapter we only experiment with 2-hop attention. However the model

can be easily extended to k-hop ($k > 2$) attention models.

4.4.4 Auxiliary Supervision

Another advantage of using multi-hop attention is that we can apply extra supervision (Hill et al., 2016) on the attention scores to force the model to learn any arbitrary attention distribution as desired. In the case of multiple implicit arguments, we want the model to attend to all missing arguments of the query event in the first hop of attention, so that the query vector receives enough information for subsequent hops. Therefore, the desired distribution has $1/k$ for all mentions of all missing arguments (assuming k mentions in total) and 0 elsewhere. For the examples in Figure 4.5, this target distribution $\mathbf{t}$ would have 0.5 for both *company-subj* and *mill-prep_to*:

$$\begin{array}{cccccccccc} & \text{build} & \text{company} & \text{plant} & \text{provide} & \text{electricity} & \text{mill} & \text{ensure} & \text{plant} & \\ \mathbf{t} = [& 0, & 0.5, & 0, & 0, & 0, & 0.5, & 0, & 0 &] \end{array}$$

Then we can add the KL-divergence between the actual attention scores in the first hop

$$\mathbf{a}' = [a'_1, a'_2, \ldots, a'_T]$$

and the desired distribution $\mathbf{t}$ to the loss function in Equation 4.2:

$$L = -\log(\max(\mathbf{a} \circ \mathbf{m_c})) + \text{kl\_div}(\mathbf{a}', \mathbf{t}) \tag{4.4}$$

4.5 Experiments

4.5.1 Implementation

Preprocessing We use the same preprocessing steps on the Wikipedia corpus as in the EVENTCOMP model (Subsection 3.5.2). Then for each paragraph, after extracting a sequence of events and a list of entities, we adopt the new setup of the argument cloze task, that is, constructing a document-query pair for every argument of every event in the sequence if the argument co-refers with at least one

argument in its preceding events (Figure 4.3c). This leads to approximately 25 million document-query pairs in the training dataset.

Initialization and Hyperparameters For training the PAR model, we initialize the embedding layer with the same event-based word2vec embeddings (see Figure 3.4) as in the EVENTCOMP model. The embedding vectors for placeholder tokens like TARGET-* and MISS-* are initialized to zero. We use a hidden size of 300 in both document encoder and query encoder, and apply a dropout layer with a rate of 0.2 on all embeddings before they are passed to the encoders. We train the model for 10 epochs with a batch size of 128, using Adagrad optimizer (Duchi et al., 2011) to minimize the negative log-likelihood loss as defined in Equation 4.2 with a learning rate of 0.01. The 2-hop PAR model is trained with the same set of hyperparameters.

4.5.2 Results on Argument Cloze

To evaluate the PAR model on the OntoNotes argument cloze task, we also need to change the way the evaluation data is created according to the new task setup (Section 4.3). This greatly reduces the number of test cases, as many cases in the original setting have the missing argument only coreferring with arguments of subsequent events, which are excluded in the new setting. Also, although now there can be more than one candidate that constitutes a correct answer to a query (as in the example in Figure 4.3c), the number of candidates also grows much larger (about three times), because we now view every argument mention rather than a whole coreference chain as a candidate. Some statistics of both the original and modified datasets are shown in Table 4.1.

We compare the PAR model to 2 baselines, the **RANDOM** baseline, which randomly selects one candidate, and the **MOSTFREQ** baseline, which selects any candidate belonging to the coreference chain with highest number of mentions. We also compare with the best performing **EVENTCOMP** model in Subsection 4.5.2.

The evaluation results are shown in Table 4.2. We can see that the PAR model outperforms the previously best EVENTCOMP model by a large margin, especially on the harder ON-LONG dataset. In the EVENTCOMP model, we find that entity salience features, that is, numbers of different types of mentions in a coreference

	ON-SHORT		ON-LONG	
	Original	Modified	Original	Modified
# doc	1027		597	
# test cases	13018	7781	18208	10539
Avg # candidates	12.06	34.99	36.95	93.89
Avg # correct	1	3.17	1	4.61

Table 4.1: Statistics of the original and modified OntoNotes argument cloze datasets.

chain, greatly improves the performance. We have also tried to add such features to the PAR model, but do not see significant improvement (sometimes adding the features even degrades the performance). This is probably due to the fact that by sequentially modeling the context through a document encoder, PAR is already encoding entity salience as some latent information in its context-aware vectors $[\mathbf{d_1}, \ldots, \mathbf{d_T}]$.

	ON-SHORT	ON-LONG
RANDOM	13.24	8.74
MOSTFREQ	35.15	26.29
EVENTCOMP	36.90	21.26
+ entity salience	46.06	31.43
PAR	**58.12**	**51.52**

Table 4.2: Evaluation results on the modified OntoNotes datasets.

To better understand why PAR is performing well, we plot the accuracy of different models on ON-LONG by the frequency of the removed argument, that is, by the number of preceding mentions referring to the argument, in Figure 4.7. We can see that entity salience boosts the performance of the EVENTCOMP model in particular for frequent entities. While PAR not only achieves comparable performance on frequent entities with EVENTCOMP + entity salience, it also maintains a relatively steady performance on rare entities, indicating that PAR is able to capture both semantic content of events and salience information of entities.

Evaluation on Multiple Implicit Arguments To test the PAR model's ability to predict multiple implicit arguments of the same predicate (Subsection 4.4.3), we

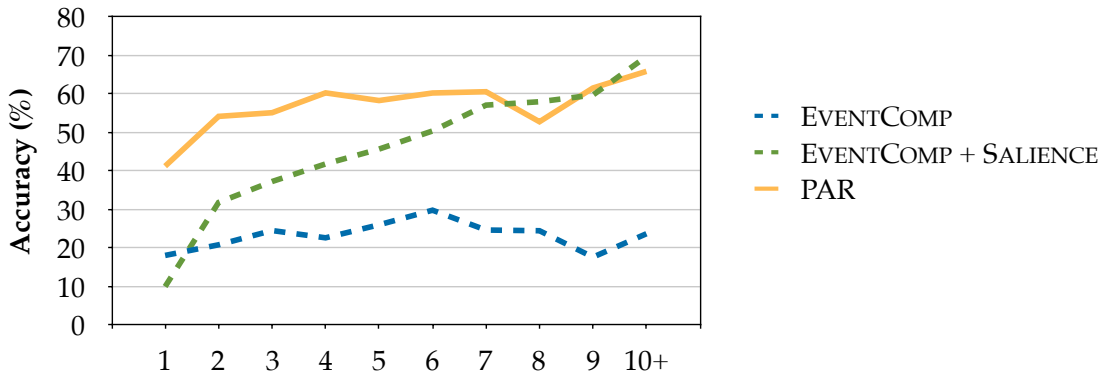


Figure 4.7: Performance of EVENTCOMP, with and without entity salience, and PAR, by entity frequency (length of coreference chain) of the removed argument, on ON-LONG.

extract subsets from both the ON-SHORT and ON-LONG datasets in which every query has more than one argument that is a potential implicit argument (i.e., co-referring with arguments of preceding events). Then we modify each query by removing all such potential implicit arguments, and ask the model to predict one of them at a time, as in the examples shown in Figure 4.5. We name the resulting two subsets ON-SHORT-MULTI and ON-LONG-MULTI.

	ON-SHORT-MULTI	ON-LONG-MULTI
PAR w/o multi-arg	51.49	43.06
PAR	48.45	39.90
2-HOP PAR	50.54	**42.69**
+ extra supervision	**50.73**	41.72

Table 4.3: Evaluating on subsets of the OntoNotes datasets with more than one missing argument in the query.

Table 4.3 shows the result of testing PAR and 2-hop PAR on the two subsets. The "PAR w/o multi-arg" evaluates PAR on the same subsets of queries, but only removes one argument at a time. The performance drop of over 3 points from the same model proves that the multi-argument cases are indeed harder than single-argument cases. The 2-hop model, however, brings the performance on multi-argument cases close to single-argument cases. This confirms our hypothesis that multi-hop attention allows the model to build a better query by reasoning over the

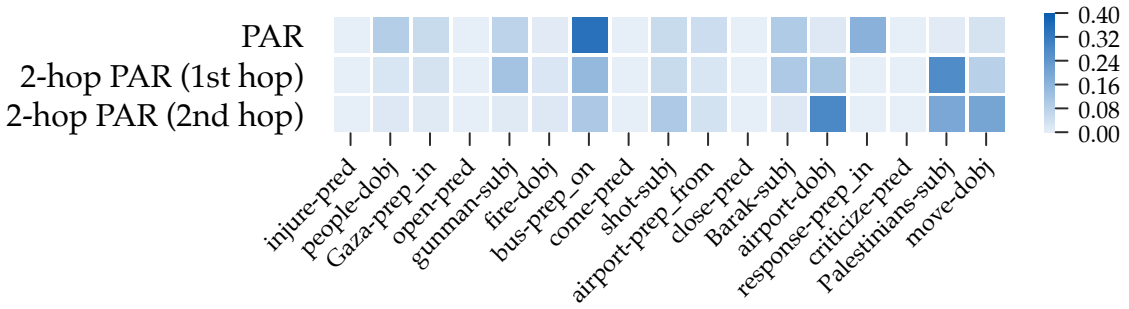
Nine people were injured in Gaza when gunmen opened fire on an Israeli bus. Witnesses say the shots came from the Palestinian international airport. Israeli Prime Minister Ehud Barak closed down the two-year-old airport in response to the incident. Palestinians criticized the move. They regard the airport as a symbol of emerging statehood.

(a) A piece of raw text from the OntoNotes corpus (`english/bn/cnn_0019`).

Document
injure-pred people-dobj Gaza-prep_in open-pred gunman-subj fire-dobj bus-prep_on come-pred shot-subj airport-prep_from close-pred Barak-subj airport-dobj response-prep_in criticize-pred Palestinians-subj move-dobj

Query
regard-pred **MISS-subj** TARGET-dobj symbol-prep_as

(b) An argument cloze example with multiple implicit arguments in the query, with the correct answers to the query marked blue in the document.



(c) The heatmap of attention scores over document input from PAR and 2-hop PAR. While PAR fails on this example, 2-hop model succeeds from a more informative query vector when the first hop attends to other missing arguments of the query.

Figure 4.8: The heatmap of attention scores on an OntoNotes example from the PAR and 2-hop PAR models.

document. We also train a 2-hop model with extra supervision on the first hop of attention scores, as discussed in Subsection 4.4.4, but it does not provide much benefit in this experiment. Figure 4.8 shows an example where PAR fails to point to the correct answer, but the 2-hop model succeeds by first attending to other missing arguments of the query (*Palestinians* as missing subject) in the first hop, then pointing to the correct answer in the second hop with a more informative query vector.

4.5.3 Results on G&C

The first obstacle of evaluating on the G&C dataset is again the fill / no-fill decision, which is particularly hard for a complex neural model to learn, as discussed in Chapter 3. Therefore, we use the same **fill / no-fill** classifier as in the previous EVENTCOMP model (see Subsection 3.5.4 for more details).

Another obstacle of applying the PAR model to G&C is, the PAR model only considers arguments of preceding verbal events (i.e., with verb predicates) as candidates. However, many of the candidates defined by the task, especially those from NomBank annotations, are not present in any verbal event (arguments of nominal predicates are likely to be absent from any dependency relation with a verb). To make a fair comparison, we convert every NomBank proposition within the candidate window to an event by mapping the nominal predicate to its verbal form, and add it to the list of preceding events. After adding the extra events, there still remains a slight difference between the candidates available to the PAR model and the candidates defined by the task, which we adjust by masking out the unavailable candidates from other models used in comparison.

Cross Validation The Wikipedia training data for PAR contains only verbal predicates, and the text is from a different domain than the G&C dataset. To bridge the gap, we fine-tune the model on G&C dataset by 10-fold cross validation, that is, for each testing fold, the model is tuned on the other nine folds. We remove the dropout layers in both document encoder and query encoder to ensure reproducibility. To prevent overfitting, we freeze the parameter weight in embedding layer and query encoder layer, using Adagrad optimizer with a learning rate of 0.0005. Still, due to the size of the dataset and the complexity of the model, the

performance is very sensitive to other hyperparameters, and we cannot find a single set of hyperparameters that works best for all models. Therefore, we report the results as an average of 5 runs with slightly different hyperparameter settings. [4]

The evaluation results are presented in Table 4.4. As a reminder, GCAUTO is an re-implementation of Gerber and Chai (2012) by removing all gold features, in order to make it a fair comparison with our methods, as discussed in Subsection 3.5.4. EVENTCOMP* evaluates the best performing EVENTCOMP model in a condition that masks out some candidates to make it a fair comparison with the PAR model, as discussed above. Note that GCAUTO, EVENTCOMP and EVENTCOMP* all have an intrinsic advantage over PAR as they exploit event information from the whole document to make the prediction, while PAR only looks at the preceding text.

	P	R	F_1
Gerber and Chai (2012)	57.9	44.5	50.3
GCAUTO	49.9	40.1	44.5
EVENTCOMP	49.3	49.9	49.6
EVENTCOMP*	48.0	48.7	**48.3**
PAR	44.0	44.7	44.4
2-HOP PAR	45.9	46.6	46.2
+ extra supervision	47.9	48.6	**48.3**

Table 4.4: Evaluating PAR on the G&C dataset.

The performance of the plain PAR model is already comparable to GCAUTO. With an additional hop of attention, the performance increases by around 2 points. This is as expected, as over 30% of the predicates in the G&C dataset have more than one implicit argument, and we have shown in Table 4.3 that multi-hop attention helps prediction on multi-argument cases. Finally, when the 2-hop model is trained with extra supervision, it gains another 1.7 points improvement, achieving an F_1 score of 48.3, on par with EVENTCOMP*, the comparably evaluated EVENTCOMP. Figure 4.9 shows the attention scores of PAR and 2-hop PAR on Example (4.1), to demonstrate the power of 2-hop inference on multi-argument cases.

[4] The hyperparameters are: $(B = 4, \lambda = 1.0)$, $(B = 8, \lambda = 1.0)$, $(B = 16, \lambda = 1.0)$, $(B = 8, \lambda = 0.1)$, and $(B = 8, \lambda = 0.0)$, where B is the batch size and λ is the ℓ_2 regularizer weight.

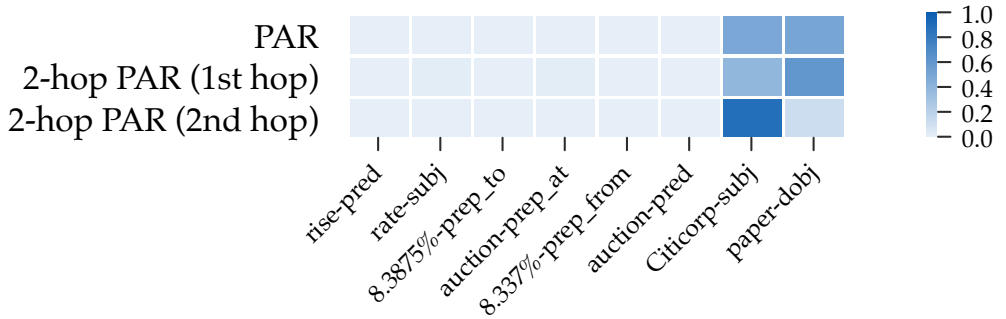
The average interest rate rose to 8.3875% at Citicorp's $50 million weekly auction of 91-day commercial paper, or corporate IOUs, from 8.337% at last week's sale.

(a) An example from the G&C dataset (same as Example (4.1)).

Document
rise-pred rate-subj 8.3875%-prep_to auction-prep_at 8.337%-prep_at auction-pred Citicorp-subj paper-dobj

Query
sale-pred TARGET-subj **MISS-dobj**

(b) The example when viewed as a document-query pair, with multiple implicit arguments in the query.



(c) The heatmap of attention scores over document input from PAR and 2-hop PAR. While the 2-hop model attends more to the non-target missing argument (paper-dobj) on the first hop, it successfully points to the target argument in the second hop.

Figure 4.9: The heatmap of attention scores on a G&C example from the PAR and 2-hop PAR models.

4.6 Chapter Summary

In this chapter, we frame implicit argument prediction as a reading comprehension task, where the predicate-argument tuple with the missing argument is a query, and the preceding text is the document in which the answer can be found. Also drawing on pointer networks and multi-hop memory networks, we introduce the Pointer Attentive Reader (PAR) model for implicit argument prediction. On an argument cloze task, PAR beats our previous best EVENTCOMP model by a large margin, showing good performance on both short and long texts, and on both salient as well as less salient arguments. When multiple arguments are missing, the use of a second hop to reason over possible arguments of the query considerably improves performance. This also proves useful on the naturally occurring implicit arguments dataset from Gerber and Chai (2010), where we also show that applying extra supervision on the first-hop attention scores further boosts performance.

Chapter 5

Semantic Structure as Supervision for Self-Attention

This chapter introduces a framework to guide self-attention computations in the Transformer architecture (Vaswani et al., 2017) with semantic structural knowledge, in the form of auxiliary supervision. The semantic knowledge we use here consists of coreference information along with predicate-argument relations. The work in this chapter has been published in Cheng and Erk (2020), where I developed the models, conducted the experiments and analysis, and wrote the paper, under the advice of Katrin Erk. All work in this chapter constitutes original contributions.

5.1 Chapter Overview

In Chapter 3 and Chapter 4, we focus on learning latent representations of events and their arguments to infer the implicit predicate-argument relations from text. Given that implicit arguments widely exist in natural language, an interesting question we want to ask next is: Would the knowledge of implicit arguments, or more broadly, *linguistic knowledge of semantic structure* (e.g., entities, events, etc), be beneficial to other downstream tasks on natural language understanding? While the models studied in the previous chapters show good performance on both synthetic and naturally-occurring data of implicit arguments, it is non-trivial to generalize them to other tasks that would benefit from the knowledge of implicit arguments, because they are deliberately designed to predict implicit arguments, and they take pre-processed event tuples instead of raw text as input. Therefore, in this chapter, we shift our focus to models that *take linguistic structure into account in a more latent manner*.

Large-scale pre-trained language models such as ELMo (Peters et al., 2018), GPT (Radford et al., 2018), BERT (Devlin et al., 2019), and XLNet (Yang et al., 2019b) have recently achieved state-of-the-art results on a wide range of NLP tasks, approaching human performance. These models, mostly built on a stacked self-attention architecture as in the Transformer (Vaswani et al., 2017), are not explicitly trained to take linguistic structure into account, but they have been shown

to encode some linguistic knowledge anyway, in particular knowledge related to syntactic structure (Tenney et al., 2019b, Jawahar et al., 2019). However, on some tasks that require complex and long-distance reasoning where surface-level cues are not enough, there is still a large gap between the pre-trained models and human performance, including tasks that require knowledge of broader discourse (Paperno et al., 2016) or coreference (Dasigi et al., 2019), or that require identifying valid reasoning (Niven and Kao, 2019). For such tasks in particular, it is interesting to test whether explicit information about linguistic structure can be helpful, how such structure should be injected, and whether it can be helpful in addition to recent language models. Strubell et al. (2018) recently showed that it is possible to inject knowledge of syntactic structure into a Transformer-style model by applying supervision on self-attention weights. Intriguingly, their model, which they apply to semantic role labeling (SRL), benefits orthogonally from the use of ELMo embeddings, and the integration of supervised self-attention, showing that syntactic knowledge and pre-trained language models can be mutually beneficial to some extent.

In this chapter, we consider a task that was explicitly designed to require long-distance knowledge, the LAMBADA task (Paperno et al., 2016). LAMBADA is a language modeling task on narrative text passages (Zhu et al., 2015), where the models are asked to predict the last word of a sentence. The test set data points are chosen to be easily solvable for humans given a larger preceding context of several (on average 4 to 5) sentences, but impossible to solve for humans given only a single sentence (see Figure 5.2 for an example). In the paper that originally introduced LAMBADA, Paperno et al. (2016) reported the highest model accuracy of only 7.3%. Since then, more recent models have improved performance to 63.24% (Radford et al., 2019, GPT-2), while human performance is above 80%. So the LAMBADA dataset clearly has the characteristic described above, with a large gap between pre-trained models and human performance. We discuss the dataset in more detail in Section 5.3.

We test whether an injection of linguistic knowledge in a similar manner to Strubell et al. (2018), i.e., adding supervised self-attention to an existing model, will improve performance on this complex task. As LAMBADA focuses on narrative texts, we hypothesize that certain semantic knowledge, including the entities

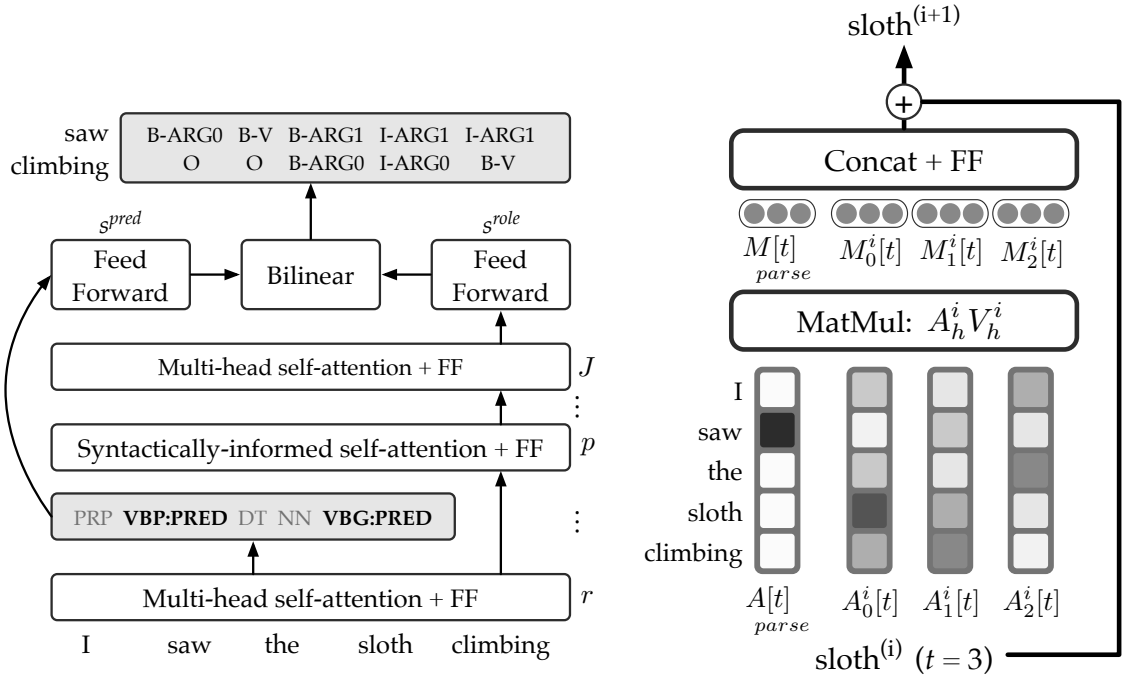
mentioned in the passage, and relations among events and their participants, will be particularly useful for solving the task (Section 5.4). We find that a BIDAF-based (Seo et al., 2017) model trained with auxiliary supervision from semantic knowledge achieves state-of-the-art performance (Section 5.5), while containing with only a tiny fraction of trainable parameters compared to the previous best model, the largest GPT-2 model. We further analyze the results in more detail (Section 5.6), finding evidence that the auxiliary supervision enables the model to better capture semantic structure in the text. To provide some insights on how to apply similar techniques to other problems, we also experiment with different model variants to test where best to insert the supervision into the system.

5.2 Prior Work

As discussed in Subsection 2.2.3, recent pre-trained language models do not explicitly take any linguistic structure into account, as the pre-training objective is to predict the next word, a randomly masked word, or the next sentence. While these pre-trained models achieved state-of-the-art results on many tasks, it is still largely unknown to what extent implicit knowledge of linguistic structure, such as syntactic structure or coreference, contributes to the improvement. Tenney et al. (2019b) designed a list of probing tasks to test how well the contextualized representations learned from ELMo / GPT / BERT do on some core NLP pipeline tasks, and found out that contextualized embeddings improve largely on syntactic tasks (like part-of-speech tagging and parsing) but not so much on semantic tasks (like coreference).

Strubell et al. (2018) recently achieved state-of-the-art performance on semantic role labeling by injecting syntactic knowledge into a Transformer-style model. In their LISA model (**L**inguistically-**I**nformed **S**elf-**A**ttention) consisting of 10 to 12 self-attention layers (Figure 5.1a), one self-attention head in one of the layers is configured to learn dependency parsing via an auxiliary supervision signal that encourages each token to only attend to its syntactic parent (Figure 5.1b). They also showed that such syntactically-informed self-attention can be combined with ELMo embeddings to further improve performance over a baseline with only ELMo and self-attention but no auxiliary supervision. In this paper, we want to

investigate whether linguistic knowledge of semantic structure can be injected in a similar manner.



(a) The overall architecture of the LISA model. (b) Syntactically-informed self-attention.

Figure 5.1: The LISA model from Strubell et al. (2018).

5.3 The LAMBADA Task

Paperno et al. (2016) introduced the LAMBADA dataset, a specially designed language modeling task where each data point is a passage composed of a context (on average 4 to 5 sentences) and a target sentence, and the task is to guess the last word of the target sentence. The data comes from narrative text in the BooksCorpus (Zhu et al., 2015), and is filtered by human subjects such that it is easy for humans to guess the target word when provided with the whole passage, but impossible to guess given only the target sentence. Figure 5.2 shows an example.

Paperno et al. (2016) also experimented some baselines and some standard language models (N-Gram, RNN, LSTM, Memory Network) on the task. The results

Context: "By the way, Elizabeth asked if I'd seen you," Tony lied. He wanted Jon to leave so he could talk with Ezekiel alone. There was something that aunt Casey, Patella and Gabriella had said about Tom that had bothered him ever since meeting Ezekiel earlier that afternoon.

Target sentence: "I'm sure she'll find me," Jon remarked curtly, trying to cut short the conversation with ___ .

Target word: Tony

Figure 5.2: An example from the LAMBADA dataset.

of the language models are extremely low as none of them reached an accuracy of 1%, while the best baseline, i.e., selecting a random capitalized word from passage, gave an accuracy of 7.3%, indicating the difficulty of the task.

Chu et al. (2017) proposed to a new evaluation setup for the LAMBADA task, by viewing it as reading comprehension, with the context sentences as the passage and the target sentence without the last word as the query. The model is then asked to select a word from the passage as the answer to the query. Despite the fact that models under this setup will decidedly fail on 19% of the test cases where the target word is not in the context, doing so still greatly boosts performance, partly because in the remaining 81% cases the model only needs to consider words appeared in the passage as candidates, rather than the whole vocabulary. Chu et al. (2017) tested several widely-used reading comprehension models, including the *Stanford Reader* (Chen et al., 2016), the *Attention Sum Reader* (Kadlec et al., 2016), and the *Gated-Attention Reader* (Dhingra et al., 2017), and achieved a best result of 49.0% with the Gated-Attention Reader plus some hand-designed features.

Dhingra et al. (2018) improved the number to 55.69% by combining the *Gated-Attention Reader* with a "Coref-GRU" layers, where both the previous token and the co-referent antecedent serve as the input to the current GRU cell. Hoang et al. (2018) combined the *Attention Sum Reader* with a multi-task objective to track entities in the context, further improving performance to 59.23%. Both these experiments proved the effectiveness of entity knowledge in the task.

There have also been some efforts in applying Transformer-style models to the

task. Dehghani et al. (2019) proposed the Universal Transformer, by tying the weights of self-attention layers across depth, and achieved an accuracy of 56.25%, in comparison to 39.88% of a standard Transformer. Most recently, Radford et al. (2019) reported 63.24% with the largest GPT-2 model (1.5 billion parameters), setting the current state-of-the-art. Nonetheless, it is still far from the human performance of 86% estimated by Chu et al. (2017).

We summarize the results from major previous work in Table 5.1.

Models		Acc. (%)
Paperno et al. (2016)	Random capitalized word from passage	7.3
	N-Gram Language Model with Cache	0.1
Chu et al. (2017)	Stanford Reader	21.7
	AS Reader	41.4
	AS Reader + features	44.5
	GA Reader	45.4
	GA Reader + features	49.0
Dhingra et al. (2018)	GA Reader	53.98[1]
	GA Reader + Coref-GRU	55.69
Hoang et al. (2018)	AS Reader	55.60[1]
	AS Reader + features + entity tracking	59.23
Dehghani et al. (2019)	Transformer	39.88
	Adaptive Universal Transformer	56.25
Radford et al. (2019)	GPT-2-small (117M parameters)	45.99
	GPT-2-medium (345M parameters)	55.48
	GPT-2-large (762M parameters)	60.12
	GPT-2-xl (1.5B parameters)	63.24

[1] These numbers from Dhingra et al. (2018), Hoang et al. (2018) are higher than their counterparts from Chu et al. (2017), probably because they applied another filtering step on their training set, which is not explicitly discussed in their papers. We elaborate more on this point in Subsection 5.5.1.

Table 5.1: Results on the LAMBADA task from major previous work. Note: AS Reader refers to Attention Sum Reader (Kadlec et al., 2016), and GA Reader refers to Gated Attention Reader (Dhingra et al., 2017).

5.4 Methods

5.4.1 Task Formulation

In this chapter, we adopt the reading comprehension setup from Chu et al. (2017), following most of the previous work on the task. Formally, we concatenate all tokens in the context sentences to get the context input $\mathbf{x} = \{x_1 \dots x_n\}$. We represent all but the last word from the target sentence as the query input $\mathbf{q} = \{q_1 \dots q_m\}$, and the last word of the target sentence as the answer a.

The model computes a probability of being the correct answer for each word in the context $P(x_i|\mathbf{x}, \mathbf{q})$. Because the answer a might occur multiple times in the context, at training time, we sum the probabilities of all correct tokens, and compute the loss as the negative log-likelihood of the summed probability:

$$\mathcal{L}_0 = -\log P(a|\mathbf{x}, \mathbf{q}) = -\log \sum_{i:x_i=a} P(x_i|\mathbf{x}, \mathbf{q}) \tag{5.1}$$

At test time, a pointer sum mechanism (Kadlec et al., 2016) is used to predict the word type with the highest summed probability among all distinct word types in the context.

5.4.2 Model

The aim of this chapter is to test whether linguistic knowledge of semantic structure can be injected into an existing model via supervised self-attention, and whether the performance of such a model on the LAMBADA task can be matched with the large-scale pre-trained language models (i.e., GPT-2).[1]

As discussed in Section 5.3, a range of different reading comprehension models (i.e., Gated-Attention Reader, Attention-Sum Reader) have been tested in the previous work, and they all showed reasonably strong performance on the task (Dhingra et al., 2017, Hoang et al., 2018). Therefore, we decide to start with a conventional reading comprehension model, and fuse into it a simpler and shallower stacked self-attention architecture (with much smaller numbers of layers, atten-

[1]We take this route, rather than adding supervision to an existing GPT-2 model, because of the high computational cost of training such a large Transformer model from scratch.

tion heads, and hidden size compared to GPT-2). We choose another widely-used reading comprehension model, the BIDAF model (Seo et al., 2017), as our starting point, because BIDAF has consistently outperformed the aforementioned models in many reading comprehension benchmarks.

BIDAF Baseline The original BIDAF model, as illustrated in Figure 5.3a, mainly consists of the following components:

1. **Embedding Layer** represents each token in the context and the query by a concatenation of GloVe embeddings (Pennington et al., 2014) and Character-CNN embeddings (Kim, 2014).

2. **Contextual Layer** encodes the context sequence and the query sequence with a bidirectional-LSTM encoder.

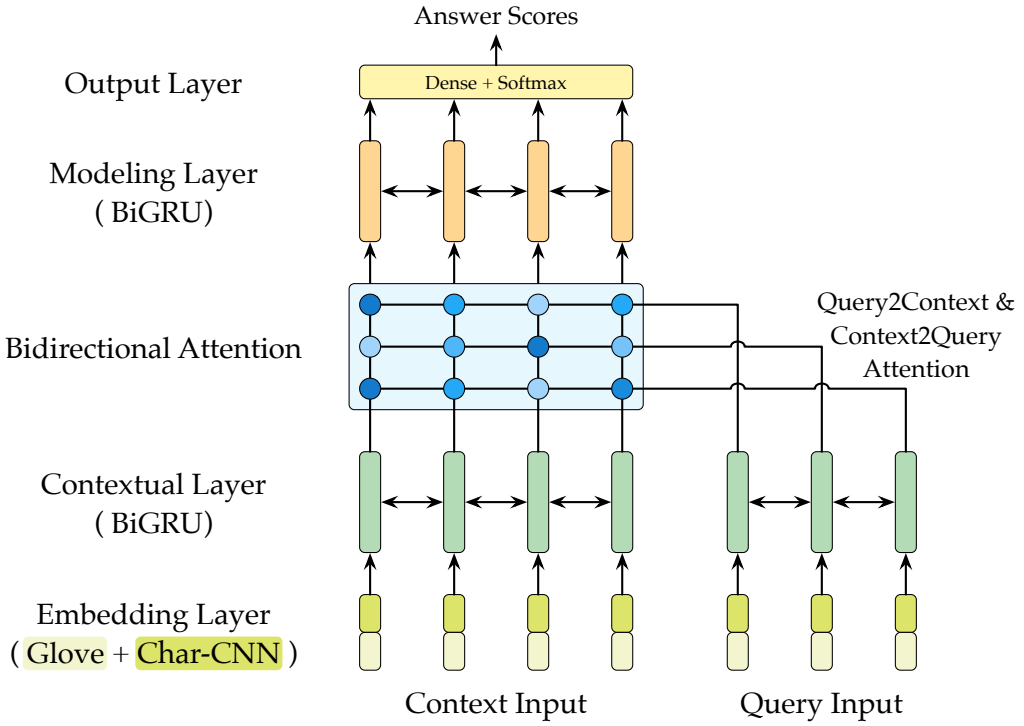
3. **Bidirectional Attention Layer** computes both context-to-query and query-to-context attentions, which are then used to merge the query representations and the context representations to get query-aware vector representations for each context word.

4. **Modeling Layer** encodes the query-aware context representation with another bidirectional-LSTM encoder to capture the interaction among context words conditioned on the query.

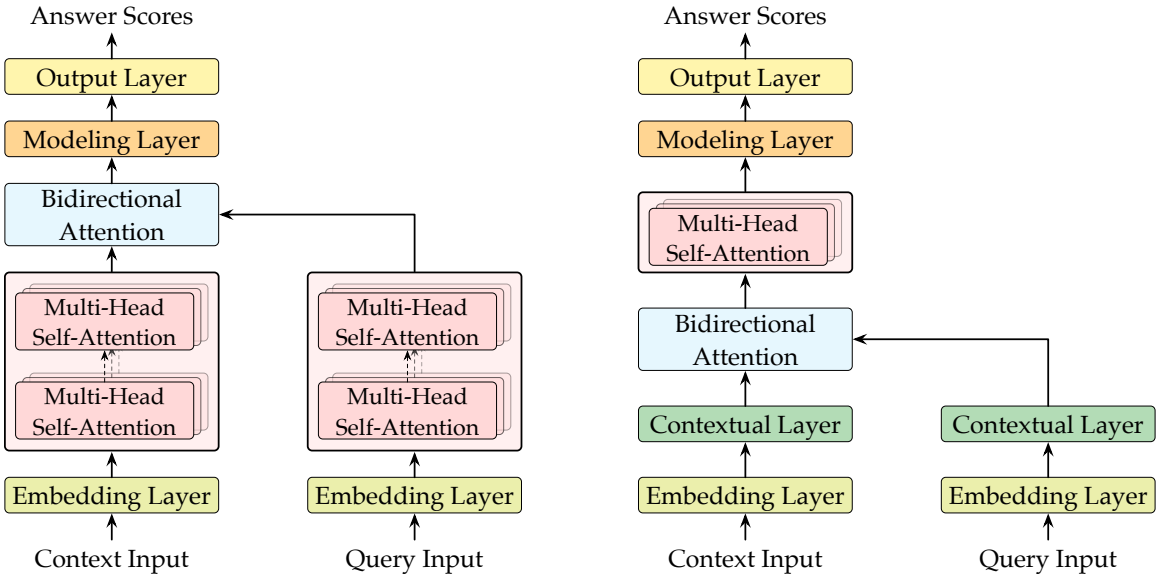
5. **Output Layer** predicts the probability for each context word being the correct answer with a feed-forward layer followed by a softmax layer.

Our baseline model is mostly the same as the original BIDAF model, except for a few small changes: we substitute the LSTMs for GRUs; we apply Layer Normalization (Ba et al., 2016) after the bidirectional attention layer and after the modeling layer to improve stability (see Section 5.5).

BiDAF with Self-Attention In order to inject semantic knowledge into the model via supervised attention, we need to fuse a stacked multi-head self-attention encoder into the BIDAF model. Intuitively, there are two options on where the self-attention encoder fits in:



(a) The baseline BIDAF model.



(b) The BIDAF-SA-EARLY variant.

(c) The BIDAF-SA-LATE variant.

Figure 5.3: The original BIDAF model (Seo et al., 2017) that we use as a baseline in our experiment, and two variants with a self-attention encoder (in red) being added either as the contextual layer or after the bidirectional attention layer.

a) Use the encoder to replace the **Contextual Layer**, as shown in Figure 5.3b.

   This is inspired by the trend of using self-attention encoders to replace traditional RNN-based encoders in many NLP problems. Also, a common practice in using pre-trained language models like BERT for downstream tasks is to first encode raw input with BERT and then pass the output to higher-level task-specific layers, which is similar to what we do here. We name this variant **BIDAF-SA-EARLY**.

b) Add the encoder after the **Bidirectional Attention Layer**, as shown in Figure 5.3c.

   This is inspired by the BIDAF++ model (Clark and Gardner, 2018), where a standard self-attention layer is added after the bidirectional attention layer to help reason over multiple paragraphs. Here we instead use multi-head self-attention, since applying auxiliary supervision on an attention layer with just one attention head leads to inferior performance in our preliminary experiments. We name this variant **BIDAF-SA-LATE**.

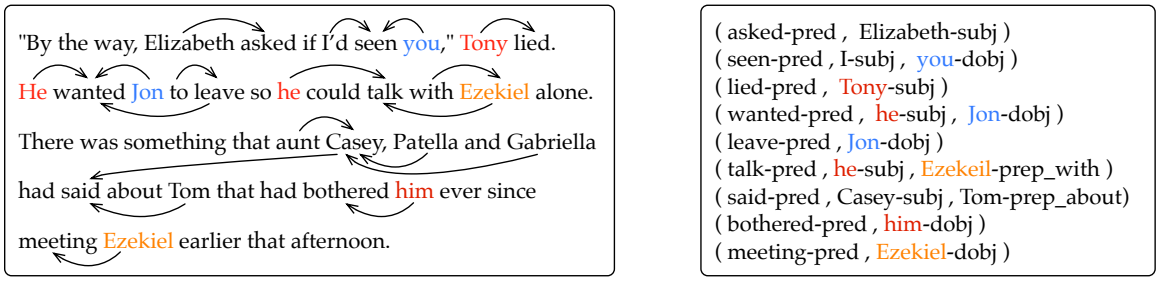
We also explore a third variant that combines the above two options, and name it **BIDAF-SA-BOTH**.

5.4.3 Auxiliary Supervision for Self-Attention

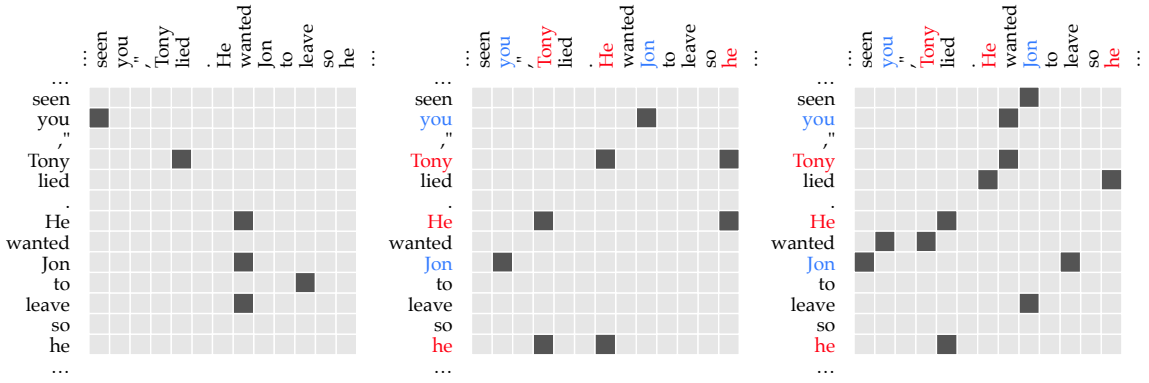
Similar to Strubell et al. (2018), we want to apply auxiliary supervision on a self-attention encoder to guide the model to learn some specific linguistic structure. Our model receives context input, namely the passage, as well as query input, which is the target sentence minus the last word. We focus on investigating auxiliary supervision on the context input, because the context input, with 4 to 5 sentences on average, should exhibit much richer linguistic structure than the query input, which is a single sentence.

To examine what kind of linguistic structures are beneficial to the problem, we experiment with 3 types of supervision signals:

Syntax Supervision: Given the dependency parses for each sentence in the context, we construct the target self-attention weights by putting a weight of 1 from



(a) **Left:** The dependency parses (arrows) and coreference chains (color-coded) of a context input (from the same example as in Figure 5.2), which are used to construct different auxiliary supervision signals. **Right:** A list of events extracted from the context, using the same heuristics as described in Section 3.3.



(b) Examples of the target self-attention weights from 3 different supervision types (only showing part of the full matrices due to space limit). Light gray represents 0, and dark gray stands for 1.

Figure 5.4: An example showing how we construct different types of supervision signals from pre-processed text input.

each token to its syntactic head token, and 0 otherwise, as shown in the left column of Figure 5.4b. We name this type of supervision **DEPPARSE**.

This is similar to the auxiliary supervision used in Strubell et al. (2018), except that we have multiple sentences rather than just one sentence. If an attention head is trained with this syntax supervision, we constrain the self-attention window by sentence boundaries, that is, each token can only attend to other tokens in the same sentence, to make it easier for the model to approach the target self-attention weights.

Coreference Supervision: Given a list of coreference chains from the context (each coreference chain contains a set of mentions that refer to the same entity), we construct the target self-attention weights by putting a weight of 1 between each pair of mention heads in the same coreference chain, and 0 otherwise, as shown in the middle column of Figure 5.4b. We name it **COREFALL**.

We also test other variants of coreference supervision, namely, guiding the head of each mention to only attend to the head of the most recent previous mention, or to the head of the immediately following mention. We refer to these two variants as **COREFPREV** and **COREFNEXT** respectively.

Narrative Supervision: Since the LAMBADA dataset is built from a corpus of novels, we hypothesize that narrative structures, that is, the sequence of events and their participants, could also be important for predicting the missing word. The interaction between the predicate and arguments of a single event is largely captured by the syntax supervision described above. Therefore, we combine the dependency parses and coreference chains to construct another type of self-attention targets that reflect higher-level narrative knowledge, as shown in the right column of Figure 5.4b: For each event argument a, we put a weight of 1 between a and all predicates that have an argument that co-refers with a. This is related to the reasoning of implicit event argument structures, in that we create the synthetic argument cloze examples in a similar way, as discussed in Section 3.3 and Section 4.3. We name this supervision **NARRATIVE**.

Note that while we require some extra information (i.e., dependency parses and coreference chains, as shown in Figure 5.4a) to construct the auxiliary supervision

signals, we do NOT rely on any gold annotations on either the training set or the test set. All the information can be obtained automatically from running existing NLP tools. We discuss the pre-processing steps more in Section 5.5.

Training with Auxiliary Supervision If any auxiliary supervision is applied to a self-attention head, we compute a loss from the supervision matrix $S \in \mathbb{R}^{n \times n}$ and the attention weights $A \in \mathbb{R}^{n \times n}$ as:

$$\mathcal{L}_s = \frac{1}{k} \sum_{i=1}^{n} \left[-\mathbb{I}_{\sum_{j=1}^{n} S_{ij} >= 0} \cdot \log \left(\sum_{j=1}^{n} A_{ij} * S_{ij} \right) \right] \tag{5.2}$$

where $\mathbb{I}_{\sum_{j=1}^{n} S_{ij} >= 0}$ is an indicator function that returns 1 when there is at least one non-zero element in the ith row of S and 0 otherwise, and k is the number of rows in S such that there is at least one non-zero element in the row. To explain, for each token with at least one supervision target, we compute the negative log-likelihood loss against all of its supervision targets, and then get the mean value.

The model is trained in an end-to-end fashion, no matter whether any auxiliary supervision is applied. To train a model with auxiliary supervisions $s_1, s_2, \ldots$, the total loss being optimized is:

$$\mathcal{L} = \mathcal{L}_0 + \lambda * \sum_i \mathcal{L}_{s_i} \tag{5.3}$$

where L_0 is the final prediction loss defined in Equation 5.1, and λ is a hyper-parameter to balance between the final prediction loss and the auxiliary losses.

5.5 Experiments

5.5.1 Dataset & Pre-processing

When introducing the LAMBADA dataset, Paperno et al. (2016) divided the BooksCorpus (consisting of 5,325 unpublished novels) randomly into 2 partitions, and only asked human subjects to filter the second half to create the development / test set, while leaving the first half raw data (2,662 novels) untouched to be the training set (thus not in the same format as development / test set). With the read-

ing comprehension setup introduced by Chu et al. (2017), they also constructed a new training set of ~1.6M instances out of the original raw data. Each training instance consists of a context with 4 to 5 sentences and a target sentence, in which the last word in the target sentence must exist in the context. Follow-up work (Dhingra et al., 2018, Hoang et al., 2018) further filtered the new training set by removing all instances where the target word is a stop word, leaving ~700k instances. This is to reduce the domain discrepancy between training and test, because only about 1% of development set instances have a stop word as the target word. We follow the latest setup here. A summary of dataset statistics is shown in Table 5.2.

	TRAIN	DEV	TEST
Size	709,568	4,869	5,153
% Answer-in-context	100%	82.4%	81.7%
Filtered by human subjects	No	Yes	Yes

Table 5.2: Statistics of the LAMBADA dataset.

As discussed in Subsection 5.4.1, we also need to obtain the dependency trees and coreference chains of the context input in order to construct the target attention weights for auxiliary supervisions. We use the neural dependency parser and the statistical coreference system from Stanford CoreNLP toolkit (Manning et al., 2014) to pre-process the whole dataset. Further discussion on the choice of pre-processing alternatives will be in Section 5.6.

5.5.2 Implementation Details

We build our models and run all the experiments with AllenNLP (Gardner et al., 2018). For the baseline BIDAF model, we mostly follow the settings of hyper-parameters of the original model:

- We use the concatenation of 100d GloVe embeddings together with 100 1D filters each with a width of 5 for Character-CNN embeddings for the embedding layer (with a combined output dimension of 200).
- We use a 1-layer BiGRU for the contextual layer, and 2-layer BiGRU for the modeling layer, each with a hidden size of 100.

- We apply a dropout with rate 0.1 on the character CNN, all BiGRU layers, and the feed-forward layer before the final prediction.
- We use a batch size of 128, and the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001.
- During training, we maintain a moving average of all model weights with a exponential decay rate of 0.9999. At test time, the moving average is used instead of the raw weights.
- We train the model for 10 epochs, and perform early-stopping when the validation accuracy does not increase for two consecutive epochs. We use validation accuracy to select the best epoch, from which then weights are then used for test set evaluation.

For the self-attention blocks in the BIDAF-SA-* variants:

- We use a size of 200 for the projection of keys, queries, and values respectively (which are then divided by the number of attention heads) in each multi-head self-attention layer, to maintain the same size on later layers as the baseline model.
- In each self-attention layer, we also apply a dropout with rate 0.1 on the attention sub-layer, the feed- forward sub-layer, and the residual connections.

In training the BIDAF-SA-* variants, we use almost the same setup as in training the baseline model, except that we also use the learning rate scheduler as described in Vaswani et al. (2017), with a warm-up step size of 8,000, as some preliminary experiments show the importance of such scheduler when using a stacked self-attention architecture:

$$lr = d_{model}^{-0.5} \cdot \min(step^{-0.5},\ step^{0.5} \cdot warmup^{-1.5}) \tag{5.4}$$

For models trained with auxiliary supervision, we use a multiplier $\lambda = 0.3$ for the auxiliary loss in Equation 5.3. We test different λ values: 0.1, 0.3, 0.5, 0.7, 1.0, and find that 0.3 works the best.

For the multi-head self-attention encoders in the BIDAF-SA-* variants, we always use 4 attention heads per layer. For BIDAF-SA-EARLY, we include 4 layers

in the stacked self-attention encoder, as preliminary studies show that using only 1 or 2 layers damages performance heavily, while using more than 4 layers gives no significant improvement. Also, in all following sections, unless otherwise specified, the auxiliary supervision is applied on the 3rd layer.[2] For BIDAF-SA-LATE, we only add 1 multi-head self-attention layer, because again, preliminary results show no further gain of using 2 or more layers.

In some experiments, we also try replacing the embedding layer with the pre-trained ELMo embeddings (Peters et al., 2018), on which we use a dropout with rate 0.2 and a projection down to 200d (to keep the same output dimension as the original word embedding layer).

We find that performance is very sensitive to the initial random state, possibly due to the fact that there is a large statistical discrepancy between the training set and the development / test sets (because the training set is not filtered by human subjects). We observe a similar effect when we re-train existing models from the literature (Dhingra et al., 2018, Hoang et al., 2018). Therefore, for each model variant, we train 4 different runs with different random seeds, and report the average and maximum performance (in parentheses in the following tables) across the 4 runs.

5.5.3 Main Results

We first experiment with the BIDAF-SA-EARLY model and the COREFALL supervision[3], because 1) intuitively, knowledge about coreference chains in the passage is likely to be the most beneficial factor for solving the task, and 2) some preliminary experiments also show consistent improvement from coreference supervision. Results from other model variants and other types of supervision signals are discussed in Section 5.6.

We compare our method with the two best previous approaches that did not use large-scale pre-training language models (Dhingra et al., 2018, Hoang et al., 2018), and the largest GPT-2 model (Radford et al., 2019). Note that we do not compare to other major pre-trained language models, because BERT (Devlin et al.,

[2]We discuss this choice in Subsection 5.6.3.

[3]The supervision is applied on the 3rd layer of the 4-layer stacked self-attention encoder, as discussed in Subsection 5.5.2

2019) and its follow-ups like XLNet (Yang et al., 2019b) and RoBERTa (Liu et al., 2019) all used the BooksCorpus as part of the pre-training data. As the LAMBADA task is constructed from the BooksCorpus, BERT and other models would gain an unfair advantage on the task because all the test instances have been accessed by these models during pre-training.

Models	Accuracy (%)
Dhingra et al. (2018)	55.69
Hoang et al. (2018)	59.23
GPT-2 (1.5B) (Radford et al., 2019)	63.24
BIDAF	59.12 (59.54)
BIDAF-SA-EARLY	60.54 (60.88)
BIDAF-SA-EARLY + COREFALL	61.51 (61.94)
BIDAF-SA-EARLY + ELMo	61.38 (61.87)
BIDAF-SA-EARLY + ELMo + COREFALL	**63.71 (64.62)**

Table 5.3: Main evaluation results on the LAMBADA test set: average accuracy across 4 runs, with max accuracy in parentheses.

Table 5.3 shows the evaluation results on the LAMBADA test set. We see that our BIDAF baseline already performs similarly to the previous best results before GPT-2. Adding the COREFALL auxiliary supervision consistently improves accuracy, with or without ELMo embeddings, but we see larger improvement from COREFALL with ELMo embeddings (~2.3 points) compared to that without ELMo embeddings (~1 point). This confirms our hypothesis that the injection of semantic knowledge via supervised self-attention can be helpful in addition to recent pre-trained language models. Also, the fact that ELMo itself only brings less than 1 point of improvement without COREFALL emphasizes the difficulty of the task. With both ELMo embeddings and COREFALL supervision, we achieve an average accuracy of 63.71% (and the best run achieves 64.62%), outperforming the largest GPT-2 model. This is quite surprising, considering that our model only contains 2.6 million trainable parameters[4], significantly smaller than the number of parameters in GPT-2 (1.5 billion).

[4]Even if we take the ELMo parameters that are not updated in our training into account, the total number of parameters is still only 96 million.

5.6 Analysis

In this section, we aim to understand why auxiliary supervision from semantic knowledge helps, what is the best possible way to apply auxiliary supervision, and how different types of supervision signals compare.

5.6.1 Does pre-processing quality affect performance?

The statistical coreference system from Stanford CoreNLP, from which we construct the supervision signals, is not the current best coreference model in terms of benchmark metrics. We also experimented with a more recent end-to-end neural coreference model (Lee et al., 2017)[5], with much higher benchmark scores, as the source of supervision for our BIDAF-SA-EARLY + COREFALL model. Surprisingly, this yields inferior performance (61.13% on average, compared to 61.51% with the Stanford coreference results).

> ***Context:*** "By the way, Elizabeth asked if I'd seen you," Tony lied. He wanted Jon to leave so he could talk with Ezekiel alone. There was something that aunt Casey, Patella and Gabriella had said about Tom that had bothered him ever since meeting Ezekiel earlier that afternoon.
>
> ***Target sentence:*** "I'm sure she'll find me," Jon remarked curtly, trying to cut short the conversation with ___ .
>
> ***Target word:*** Tony ***Prediction (Wrong):*** Ezekiel

Figure 5.5: Does pre-processing quality affect performance? An example where a wrong coreference chain (color-coded) from a neural coreference system (which we do not use for our experiments) leads to a wrong prediction. A better coreference output from the Stanford coreference system on this example is shown in Figure 5.4a.

We manually examined the output of both coreference systems on some data points, and found that the neural coreference system often produces highly erroneous output, possibly because it is overfitting on its news-centric training data,

[5]We use a re-implementation from AllenNLP.

the OntoNotes dataset (Hovy et al., 2006), while LAMBADA consists of narrative texts. Figure 5.5 shows an example where a wrong coreference chain from the neural system leads to a wrong prediction. This is the same example as in Figure 5.4a, which shows the coreference output from the Stanford system. In this example, it is hard to predict the right answer without knowing that "you" refers to "Jon" and "he" refers to "Tony", both of which are predicted correctly by the Stanford system.

This indicates that a better coreference signal could lead to even better results on the task. We leave it to future work, given some very recent work that further improved coreference performance (Joshi et al., 2020, Wu et al., 2020).

5.6.2 Does CorefAll really learn coreference knowledge?

We want to know whether the improvement from CorefAll supervision is because the supervision will actually allow the model to better learn coreference structures, or due to some unknown confounding factors. Chu et al. (2017) manually analyzed 100 random instances from the LAMBADA development set to identify the type of reasoning needed for humans to make the right prediction, and found that 21 out of the 100 instances require coreference resolution. We test our models on these 21 instances. To obtain a larger set of instances to inspect the model, we also compare the cases in the development set where the target word is a noun to cases where the target is a pronoun, and we compare the cases where the target word is a PERSON to the cases where it is not a named entity (the most common named entity type is PERSON, with all other types occurring very rarely, so we focus on PERSON).[6]

The results in Table 5.4 show that not only does CorefAll supervision improve accuracy on the 21 instances manually classified as requiring coreference, it also strongly boosts performance on the "Pronoun" and "PERSON" subsets, in comparison to their "Noun" and "Not NE" counterparts. Though not a direct proof, this intuitively supports the claim that the auxiliary supervision does enable the model to better capture coreference information, which is likely to help the reasoning particularly over pronouns and named entities.

[6]Both the part-of-speech tags and named entity tags come from the Stanford CoreNLP toolkit.

dev subset	# instances	no supervision	with supervision
Require coref	21	48.5 ± 4.1	$\mathbf{62.0 \pm 3.5}$
Noun	2,006	58.42 ± 0.32	59.48 ± 0.23
Pronoun	2,138	72.31 ± 0.15	$\mathbf{76.72 \pm 0.53}$
Not NE	2,848	54.15 ± 0.21	54.90 ± 0.33
PERSON	1,646	72.71 ± 0.08	$\mathbf{77.95 \pm 0.66}$

Table 5.4: Does COREFALL learn coreference? Accuracy on some dev subsets, BIDAF-SA-EARLY + ELMo model with and without COREFALL supervision. We report average and standard deviation across 4 runs.

5.6.3 Where should the supervision be applied?

Is it more beneficial to apply auxiliary supervision at an earlier stage of the model, i.e., at the contextual layer as in the BIDAF-SA-EARLY model, or at a later stage, i.e., after the bidirectional attention layer as in the BIDAF-SA-LATE model? We compare performance using COREFALL supervision. Also, to disentangle the effect of architectural change, we experiment with the BIDAF-SA-BOTH model with supervision being added at different stages.

Models	Accuracy (%)
BIDAF-SA-EARLY	60.54 (60.88)
BIDAF-SA-EARLY + COREFALL	61.51 (61.94)
BIDAF-SA-LATE	59.48 (59.58)
BIDAF-SA-LATE + COREFALL	61.19 (61.54)
BIDAF-SA-BOTH	60.88 (61.27)
BIDAF-SA-BOTH + COREFALL (early)	61.72 (62.35)
BIDAF-SA-BOTH + COREFALL (late)	61.54 (61.67)

Table 5.5: Early supervision vs. late supervision: Accuracy (average of 4 runs, with max in parentheses) on the LAMBADA test set for different locations to fuse in the self-attention encoder and to apply auxiliary supervision.

Table 5.5 shows that without supervision, BIDAF-SA-EARLY offers much better results than BIDAF-SA-LATE. Although adding supervision to BIDAF-SA-LATE leads to a larger relative improvement, applying supervision at an earlier stage still leads to a better absolute performance than doing so at a later stage, which

is also confirmed by the numbers on BIDAF-SA-BOTH. This is not surprising, as intuitively coreference information about the context input should be beneficial to getting better query-aware context representations, rather than the other way around.

As discussed earlier, all our experiments with BIDAF-SA-EARLY have the auxiliary supervision applied on the 3rd layer (out of all 4 layers). We also test applying supervision on other layers, and show the results in Table 5.6. We find the 3rd layer generally works the best, though the difference is not significant. One possible explanation is: The hidden states from early layers might not encode enough contextual information for the self-attention computation to learn the target weights from the supervision; while the output of the top layer is fed directly into the bidirectional attention layer to model the interaction between the context and the query, so enforcing the self-attention computation to comply with coreference knowledge solely from the context might have a side effect on this interaction. Therefore, an intermediate layer from the stacked self-attention encoder might be the best place to apply auxiliary supervision.

Models	Accuracy (%)
BIDAF-SA-EARLY	60.54 (60.88)
BIDAF-SA-EARLY + COREFALL (1st layer)	61.27 (61.83)
BIDAF-SA-EARLY + COREFALL (2nd layer)	61.33 (61.48)
BIDAF-SA-EARLY + COREFALL (3rd layer)	**61.51 (61.94)**
BIDAF-SA-EARLY + COREFALL (4th layer)	61.45 (61.61)

Table 5.6: Which layer to apply auxiliary supervision: Accuracy (average of 4 runs, with max in parentheses) on the LAMBADA test set from BIDAF-SA-EARLY + COREFALL with auxiliary supervision applied on each layer of the stacked self-attention encoder.

5.6.4 Are other types of supervision also useful?

We have so far focused on the COREFALL supervision. In Table 5.7 we show the results of applying other types of auxiliary supervision.

All other types of auxiliary supervision, except for NARRATIVE, show inferior performance compared to COREFALL. This is as expected. As the LAMBADA task

Models	Accuracy (%)
BIDAF-SA-EARLY	60.54 (60.88)
BIDAF-SA-EARLY + COREFALL	61.51 (61.94)
BIDAF-SA-EARLY + DEPPARSE	61.06 (61.34)
BIDAF-SA-EARLY + COREFPREV	60.94 (61.71)
BIDAF-SA-EARLY + COREFNEXT	61.27 (61.63)
BIDAF-SA-EARLY + NARRATIVE	61.86 (62.39)

Table 5.7: Supervision types: Accuracy on the LAMBADA TEST set (average of 4 runs, with max in parentheses) with different types of auxiliary supervision.

is specifically designed to require broader discourse context, intra-sentence syntactic structure (DEPPARSE) should not play an important role. The COREFPREV and COREFNEXT variants of coreference information only provide guidance towards the immediately preceding or following mention in the same coreference chain. Such knowledge will fall short when the reasoning over a long coreference chain is crucial in making the prediction.

The NARRATIVE supervision provides slightly better performance than COREFALL. This is also not surprising, as the NARRATIVE signal is derived from both dependency parses and coreference chains. Theoretically, this type of supervision should capture useful linguistic structures from both COREFALL, which makes the main contribution to the performance improvement, and DEPPARSE, which might offer some additional boost. We further verify the hypothesis by computing the agreement between the predictions of two models on the dev set, and find that on average, a run with COREFALL and a run with NARRATIVE agree on 89.3% of all dev instances, confirming that it is largely the coreference signal that leads to the performance improvement observed with the NARRATIVE supervision. We also briefly test combining multiple supervision signals, and find that COREFALL + NARRATIVE leads to slightly better performance (with an average accuracy of 62.05%) than using NARRATIVE alone, but the difference is not significant as well.

5.7 Chapter Summary

In this chapter we investigate whether the injection of semantic knowledge into an existing model (BIDAF) via supervised self-attention can lead to better performance on tasks requiring complex and long-distance reasoning. On the LAMBADA dataset, where the current best result from GPT-2 (Radford et al., 2019) is still far below human performance, we show that a BIDAF model trained with coreference as auxiliary supervision achieves the new state-of-the-art, while requiring only a tiny fraction of the parameters in GPT-2. We further test model variants to test where in a BIDAF model it is most useful to add a self-attention layer with supervision, and how different types of linguistic knowledge compare as supervision signals.

Chapter 6

Semantic Knowledge on Pre-trained Language Models

This chapter introduces ongoing work on integrating semantic knowledge into large-scale pre-trained language models. We explore several different approaches of integrating semantic knowledge, and present some preliminary findings. We also discuss possible future directions. All work in this chapter has not been published and constitutes original contributions.

6.1 Chapter Overview

Pre-trained language models (LMs) have recently made impressive progress on many NLP tasks. For example, on the GLUE benchmark (Wang et al., 2019b), RoBERTa (Liu et al., 2019) achieves an average score of 88.1, outperforming the estimated human performance of 87.1. On the SQuAD 1.1 reading comprehension dataset (Rajpurkar et al., 2016), XLNet (Yang et al., 2019b) achieves an F_1 score of 95.1, well above human performance (91.2). However, this does not imply that these models possess the power of understanding human language, as several recent studies on adversarial attacks show that most of the high performance is likely due to dataset biases and the models memorizing superficial patterns:

- Jia and Liang (2017) introduce the **SQuAD Adversarial** dataset, by appending an adversarial distracting sentence to the paragraph, and find that the performance of sixteen published models at the time all drops significantly on the adversarial test set, from 75 to 36 on average. Large-scale pre-trained LMs are also not robust against this attack, as Yang et al. (2019c) report the performance of BERT (Devlin et al., 2019) drops from 91.4 on SQuAD 1.1 to 61.0 on SQuAD Adversarial.

- Wallace et al. (2019) study **universal adversarial triggers**, by using gradient-guided search to find an input-agnostic sequence of tokens that triggers a model to produce a certain output when concatenated to any input data. For example, they find that a specific sequence of six sub-word tokens will trigger a GPT-2 model (Radford et al., 2019) to always generate racist outputs.

> They also find that on the SQuAD dataset, concatenating a sequence of "`why how because [adversarial answer span]`" will often fool an ELMo-based model (Peters et al., 2018) to predict `[adversarial answer span]` as the answer to any why questions.

These findings suggest that pre-trained LMs might not be fully capable of reasoning over the semantics of language. Since these models are usually pre-trained with some variants of the language modeling objective on a gigantic corpus, their success might as well be attributed to memorizing the distributional characteristics of language. In Chapter 5, we observe that integrating coreference knowledge as auxiliary supervision for self-attention can benefit tasks requiring long-distance reasoning. Experiments show that a relatively small transformer encoder trained with coreference supervision can outperform GPT-2, a much larger pre-trained LM. Therefore, a natural question to ask next is, whether semantic knowledge can be integrated into pre-trained LMs in a similar fashion to learn representations that are more "semantic-aware", and as a result, further benefit downstream tasks.

To answer this question, we first need to choose some benchmark datasets to evaluate the models' performance before and after injecting semantic knowledge. This is non-trivial, as on many existing natural language understanding datasets, the best performance of pre-trained models is already close to or even surpassing human performance. We hypothesize that semantic knowledge is unlikely to help in such cases, because these datasets often exhibit some level of annotation biases (Gururangan et al., 2018, Geva et al., 2019, Gardner et al., 2020) that allow models to make simple predictions from surface-level cues or distributional statistics. Therefore, instead of studying the effect of semantic knowledge on any specific dataset, we ask the following question: *Does the integration of semantic knowledge make pre-trained LMs more generalizable across different datasets?* Theoretically, the better a model does on out-of-domain datasets that it has not been trained on, the less likely that it relies merely on domain-specific biases to make correct predictions.

In this chapter, we experiment with the recent MRQA 2019 Shared Task (Fisch et al., 2019), a collection of 18 question answering datasets designed to test how well MRQA (machine reading for question answering) systems can generalize to new domain. This is exactly what we need: a diversified testbed to study whether

semantic knowledge can make pre-trained LMs more generalizable. We discuss the task setup in more detail in Section 6.3.

We explore three variants of integrating semantic knowledge: as edge probing instances in a multi-task learning objective; as a target attention matrix for supervised self-attention; or as a semantically-informative masking strategy for language modeling. In preliminary experiments using the knowledge of SRL and coreference, we find that none of the three methods can provide consistent and significant improvement across several variants of the BERT model. We also discuss possible defects of the current methods.

6.2 Prior Work

We have briefly reviewed existing work on integrating knowledge into neural architectures in Section 2.3. Here, we focus on integrating **semantic knowledge** into pre-trained LMs, and thus elaborate some related work in more detail.

Probably the most relevant prior work is LIBERT (Lexically-informed BERT, Lauscher et al., 2019), that adds a third pre-training objective to BERT (Devlin et al., 2019) to predict lexical relations between input tokens. The lexical relations, such as synonyms and hypernym-hyponym pairs, are extracted from WordNet (Miller, 1995). Experiments show that LIBERT slightly outperform BERT on the GLUE benchmark. Another loosely related work is SemBERT (Zhang et al., 2020), who concatenate the BERT embedding for each token with a learned embedding for the token's semantic role labels, which is predicted by an off-the-shelf parser. However, the usage of semantic information in this model, that is, vector concatenation at the last layer without changing any behavior of the pre-trained BERT, is at a relatively superficial level, thus we do not consider it as truly "integration of semantic knowledge".

On a broader sense, there has been more prior work on integrating **background knowledge** into pre-trained LMs. One line of work is to retrieve relevant knowledge elements from a background KB to assist machine reading comprehension. Yang et al. (2019a) enhance the BERT embedding of each input token with an attention mechanism over concept embeddings learned from WordNet (Miller, 1995) and NELL (Mitchell et al., 2015), and show improved performance on tasks like

SQuAD and ReCoRD (Zhang et al., 2018). Also on the ReCoRD dataset, Qiu et al. (2019) extract a sub-graph for each input token from WordNet and ConceptNet (Speer et al., 2017) and then use graph attention networks (Veličković et al., 2018) to update the token embedding with information from the sub-graph. Similarly, Lin et al. (2019) ground each question-answer pair in the CommensenseQA dataset (Talmor et al., 2019) to a "schema graph" extracted from ConceptNet, and perform joint encoding of BERT and the schema graph to derive the plausibility score of the answer.

Another line of work has been on extending the pre-training of BERT to integrate the knowledge of real-world entities. Zhang et al. (2019b) propose the ERNIE model, by adding a knowledgeable encoder on top of the BERT contextualized encoder to incorporate entity embeddings trained on Wikidata. The entity embeddings are obtained by aligning input tokens to Wikipedia entities. In addition to the MLM and NSP objectives used in BERT (see Subsection 2.2.3), ERNIE also adds a *"denoising entity auto-encoder"* objective to predict masked token-to-entity alignments. The KnowBERT model (Peters et al., 2019a) applies a KAR (Knowledge Attention and Recontextualization) mechanism to one of the self-attention layers in BERT, by first retrieving a set of relevant entities for each token and then updating the token representations by an attention computation over the entity embeddings. The WKLM model (Xiong et al., 2020) adds a new *"entity replacement"* pre-training objective to BERT, by randomly replacing an entity mention that can be linked to a Wikipedia entity with mentions of other entities that have the same entity type, for example, from *"... comic books published by Marvel Comics"* to *"... comic books published by DC Comics"*. All these enhanced models outperform standard BERT on a number of downstream tasks like question answering, relation classification, and entity typing.

6.3 The MRQA 2019 Shared Task

The MRQA 2019 Shared Task (Fisch et al., 2019) is designed to evaluate the generalization capabilities of reading comprehension models on out-of-domain data. The organizers collect 18 distinct question answering datasets, including SQuAD (Rajpurkar et al., 2016), TriviaQA (Joshi et al., 2017), DROP (Dua et al., 2019), RACE

(Lai et al., 2017), MCTest (Richardson et al., 2013), QAMR (Michael et al., 2018), and so on. These datasets vary in **sources** (Wikipedia, news articles, web snippets, etc), **question styles** (entity-centric, relational, quantitative reasoning, etc), **annotators** (crowdworkers, domain experts, etc), and **annotation schemes** (writing questions based on passages, or retrieving passages from independently-written questions), providing a relatively broad coverage on real-world question answering scenarios.

The datasets are partitioned into three splits:

- **Split I**: 6 datasets for model training and development, but not included in evaluation.
- **Split II**: 6 datasets for model development but not for training. There is also a hidden test portion for each of the dataset in this split.
- **Split III**: 6 datasets only with hidden test portions for evaluation. They are not available for model training and development.

Participants can use Split I and the development portions of Split II to train their models, which will be submitted and evaluated on the hidden test portions of the 12 datasets in Split II and Split III, using both EM (exact match) and F_1 as evaluation metrics.

All of the datasets are adapted to a unified format of extractive reading comprehension: Given a passage $\mathbf{p} = \{p_1, p_2, \ldots, p_n\}$ and a question $\mathbf{q} = \{q_1, q_2, \ldots, q_m\}$, the model predicts a continuous span $(i, j) = \{p_i, p_{i+1}, \ldots, p_j\}$ from the passage as the answer. We will use the following example from SQuAD (Rajpurkar et al., 2016) as a running example throughout the chapter:

(6.1)
Passage: The Broncos took an early lead in Super Bowl 50 and never trailed. Newton was limited by Denver's defense, which sacked him seven times and forced him into three turnovers, including a fumble which they recovered for a touchdown. Denver linebacker Von Miller was named Super Bowl MVP, recording five solo tackles, 2½ sacks, and two forced fumbles.
Question: Which team held the scoring lead throughout the entire game?
Answer: Broncos

	Dataset	# Train	# Dev	# Test
Split I	SQuAD	86,588	10,507	–
	NewsQA	74,160	4,212	–
	TriviaQA	61,688	7,785	–
	SearchQA	117,384	16,980	–
	HotpotQA	72,928	5,904	–
	Natural Questions	104,071	12,836	–
Split II	BioASQ	–	1,504	1,518
	DROP	–	1,503	1,501
	DuoRC	–	1,501	1,503
	RACE	–	674	1,502
	RelationExtraction	–	2,948	1,500
	TextbookQA	–	1,503	1,508
Split III	BioProcess	–	–	219
	ComplextWebQuestions	–	–	1,500
	MCTest	–	–	1,501
	QAMR	–	–	1,524
	QAST	–	–	220
	TREC	–	–	1,021

Table 6.1: Overview of the datasets in the MRQA Task, from Fisch et al. (2019).

Datasets that are originally multi-choice are converted to this extractive format by matching the correct answer string to a passage span and discarding questions where the correct answer cannot be matched. Also, only the first 800 tokens of the passage are kept to alleviate computational requirements[1].

Table 6.1 provides an overview of the statistics of the 18 datasets. For more information about the datasets and setups, we refer readers to the task description paper (Fisch et al., 2019) and their GitHub repository at https://github.com/mrqa/MRQA-Shared-Task-2019.

The task organizers also implement two BERT baselines (Devlin et al., 2019), one with $\text{BERT}_{\text{BASE}}$, and the other with $\text{BERT}_{\text{LARGE}}$, which we will discuss in more detail in Section 6.4. There are ten participating systems in the shared task, most of which are developed on top of the official baseline model. The best performing system, D-Net (Li et al., 2019), adopts other pre-trained LMs including XLNet (Yang et al., 2019b) and ERNIE (Zhang et al., 2019b). Their submitted system is

[1]QA pairs where the answer cannot be located in the first 800 tokens are discarded.

an ensemble of an XLNet-based model and an ERNIE-based model. The second best performing system, Delphi (Longpre et al., 2019), explores different data sampling strategies and data augmentation techniques. Their submitted system is an XLNet-based model trained with additional negative sampling (examples where no answer exists in the passage).

Participants have also tried some other enhancements, for example, multi-task learning with natural language inference and paragraph re-ranking (Takahashi et al., 2019, Li et al., 2019), or adversarial training to learn domain-invariant features (Lee et al., 2019), but find that these techniques only provide marginal improvement. The most significant factor on performance is the choice of pre-trained LMs, as models using XLNet as the base LM consistently outperform models using BERT as the base LM. The evaluation results of the two best performing systems and two official baselines are summarized in Table 6.2.

	Split I	Split II	Split II	Split III
	Dev (6)	Dev (6)	Test (6)	Test (6)
D-Net (Li et al., 2019)	**84.1**	**69.7**	**68.9**	**76.1**
Delphi (Longpre et al., 2019)	82.3	68.5	66.9	74.6
$\text{BERT}_{\text{LARGE}}$ (Fisch et al., 2019)	76.3	57.1	57.4	66.1
$\text{BERT}_{\text{BASE}}$ (Fisch et al., 2019)	74.7	54.6	54.6	62.4

Table 6.2: Performance of the two best performing participating systems and two official baselines on MRQA. Numbers are reported as average F_1 scores on certain portions of all datasets in a split.

6.4 Methods

Like most participating systems on the shared task, we also adopt the official baseline as our starting point. We first describe the baseline model, then describe several variants of integrating semantic knowledge into the model.

6.4.1 Baseline Model

The baseline model is simply a BERT-based QA model. To illustrate, the question $\mathbf{q} = \{q_1, \ldots, q_m\}$ and the passage $\mathbf{p} = \{p_1, \ldots, p_n\}$ are first concatenated into a

single sequence **x**:

$$x_{1:T} = \{[\text{CLS}], q_1, \ldots, q_m, [\text{SEP}], p_1, \ldots, p_n, [\text{SEP}]\} \tag{6.1}$$

Note that since BERT takes sub-word tokens (i.e., word pieces) as input, here $x_{1:T}$ also represent word pieces rather than whole words. The input sequence for Example (6.1) would be:

(6.2) [CLS] who was the super bowl 50 mvp ? [SEP] the broncos took an early lead in super bowl 50 and never trailed . newton was limited by denver ' ##s defense , which sacked him seven times and forced him into three turnover ##s, ...

The sequence is encoded by BERT to obtain a list of contextualized representations:

$$\{\mathbf{y}_1, \ldots, \mathbf{y}_T\} = \text{BERT}(\{x_1, \ldots, x_T\}) \tag{6.2}$$

Then, two separate MLP classifiers are applied on the contextualized representations of all passage tokens $\{\mathbf{y}_{e_1}, \ldots, \mathbf{y}_{e_n}\}$ ($e_1, \ldots, e_n$ are the corresponding indices of passage tokens $p_1, \ldots, p_n$ in the input sequence $x_{1:T}$) to predict the start and end indices independently:

$$\begin{aligned} p_{start}(i) &= \text{softmax}(\text{MLP}_1(\mathbf{y}_i)), \qquad i = e_1, \ldots, e_n \\ p_{end}(j) &= \text{softmax}(\text{MLP}_2(\mathbf{y}_j)), \qquad j = e_1, \ldots, e_n \end{aligned} \tag{6.3}$$

During training, the BERT parameters and the MLP parameters are jointly optimized to maximize the log-likelihood of the first occurrence of the correct answer. Denoting the correct answer span as (a_i, a_j), the loss function can be written as:

$$\mathcal{L} = -\log p_{start}(a_i) - \log p_{end}(a_j) \tag{6.4}$$

At test time, the start and end probabilities are combined and a greedy search is applied to decode the answer span with the highest combined probability, by optimizing

$$\text{argmax}_{i,j}\{p_{start}(i) \times p_{end}(j)\} \tag{6.5}$$

Because BERT supports a sequence length up to 512, longer sequences are con-

verted into multiple chunks using a sliding window. Also, following the MultiQA model (Talmor and Berant, 2019), the baseline model is trained with a multi-task objective, by sampling up at most 75k examples from each of the Split I training datasets and iterating through them to create mixed batches of examples (so each batch can contain examples from different datasets).

6.4.2 Integrating Semantic Knowledge

In this chapter, we focus on two types of semantic knowledge: semantic role labeling (**SRL**), and coreference resolution (**Coref**). To get these additional annotations, we pre-process the MRQA datasets with existing tools for SRL and coreference. Note that while the performance of SRL and coreference tools has improved a lot recently, the automatic annotations are still noisy, especially on the domain of web text that these tools are not trained on. Nevertheless, we still expect the noisy annotations to provide helpful semantic information.

We explore three alternatives of integrating semantic knowledge into the baseline model: **EP** (Edge Probing), **SA** (Supervised Attention), and **LM** (Semantically-Informative Language Modeling). Before diving into the detail, we first define some common notations. Given an input sequence $[x_1, \ldots, x_T]$, denote the hidden states after layer ℓ as $\mathbf{H}^{(\ell)} = [\mathbf{h_1}^{(\ell)}, \ldots, \mathbf{h_T}^{(\ell)}] \in \mathbb{R}^{T \times d}$, where d is the hidden dimension. We also denote the self-attention weights of the h-th head in layer ℓ as $\mathbf{A}^{(\ell,h)} \in \mathbb{R}^{T \times T}$.

Also, the SRL knowledge for each example is a list of tuples $\{S_i\}$, each tuple representing a predicate-argument relation as $S_i = (s_{p1}, s_{p2}, s_{a1}, s_{a2}, s_l)$, where s_{p1} and s_{p2} are the start and end indices of the predicate span[2], s_{a1} and s_{a2} are the start and end indices of the argument span, and s_l is the semantic role label. On Example (6.2), some of the SRL tuples would be:

$(12, 12, 10, 11, \text{ARG0}) \rightarrow$ "the broncos" is the ARG0 of "took"

$(12, 12, 13, 19, \text{ARG1}) \rightarrow$ "an early lead in super bowl 50" is the ARG1 of "took"

Similarly, the coreference knowledge for each example is a list of set $\{C_i\}$, each

[2]Current SRL tools usually predict a single word as the predicate. However, a single word might be tokenized into multiple word pieces as required by BERT. Therefore, we still represent a predicate as a span to avoid ambiguity.

set representing a coreference cluster as $C_i = \{(m_1^1, m_1^2), (m_2^1, m_2^2), \dots\}$, where m_k^1 and m_k^2 are the start and end indices of a mention in the cluster. On Example (6.2), some coreference clusters would be:

$\{(10, 11), (28, 28)\} \rightarrow$ "the broncos" and "denver" are coreferent

$\{(24, 24), (35, 35)\} \rightarrow$ "newton" and "him" are corefent

Now we describe the three alternatives of integrating semantic knowledge, with a graphical illustration provided in Figure 6.1.

Edge Probing (EP) An intuitive way to integrate semantic knowledge is to ask the model to predict semantic relations as an auxiliary task. Partly inspired by a line of previous work by Tenney et al. (2019b;a), who design a suite of "edge probing" tasks to study how well do BERT embeddings capture the linguistic structure of the input text, we create similar edge probing tasks for SRL and coreference as a multi-task learning objective.

The probing task takes a left span s_1 and a right span s_2 as input, and predicts a label l between the two spans. For both spans, we aggregate token embeddings at layer ℓ via a self-attentive pooling mechanism (Lee et al., 2017) to get the span representations $\mathbf{s}_1^{(\ell)}, \mathbf{s}_2^{(\ell)}$. The two span representations are then fed into an MLP to predict the edge label. For SRL, the probing task is straightforward, as each SRL tuple S_i naturally resembles an edge probing instance, by viewing the predicate span as s_1, the argument span as s_2, and the semantic role as the target label. For coreference, we sample positive and negative mention pairs as the input. In positive pairs, the two mention spans are from the same coreference cluster, while in negative pairs, they come from different clusters. The target label is thus 1 for positive pairs and 0 for negative pairs. With a list of edge probing instances $\{s_1^{(k)}, s_2^{(k)}, l^{(k)}\}_{k=1}^K$, we compute the cross-entropy loss by:

$$\mathcal{L}_{EP} = \frac{1}{K} \sum_{k=1}^{K} \text{cross\_entropy} \left(MLP(\mathbf{s}_1^{(\ell,k)}, \mathbf{s}_2^{(\ell,k)}), l^{(k)} \right) \tag{6.6}$$

Supervised Attention (SA) Another alternative is to use semantic knowledge as a supervision signal for self-attention weights $\mathbf{A}^{(\ell,h)}$. This is a direct extension of

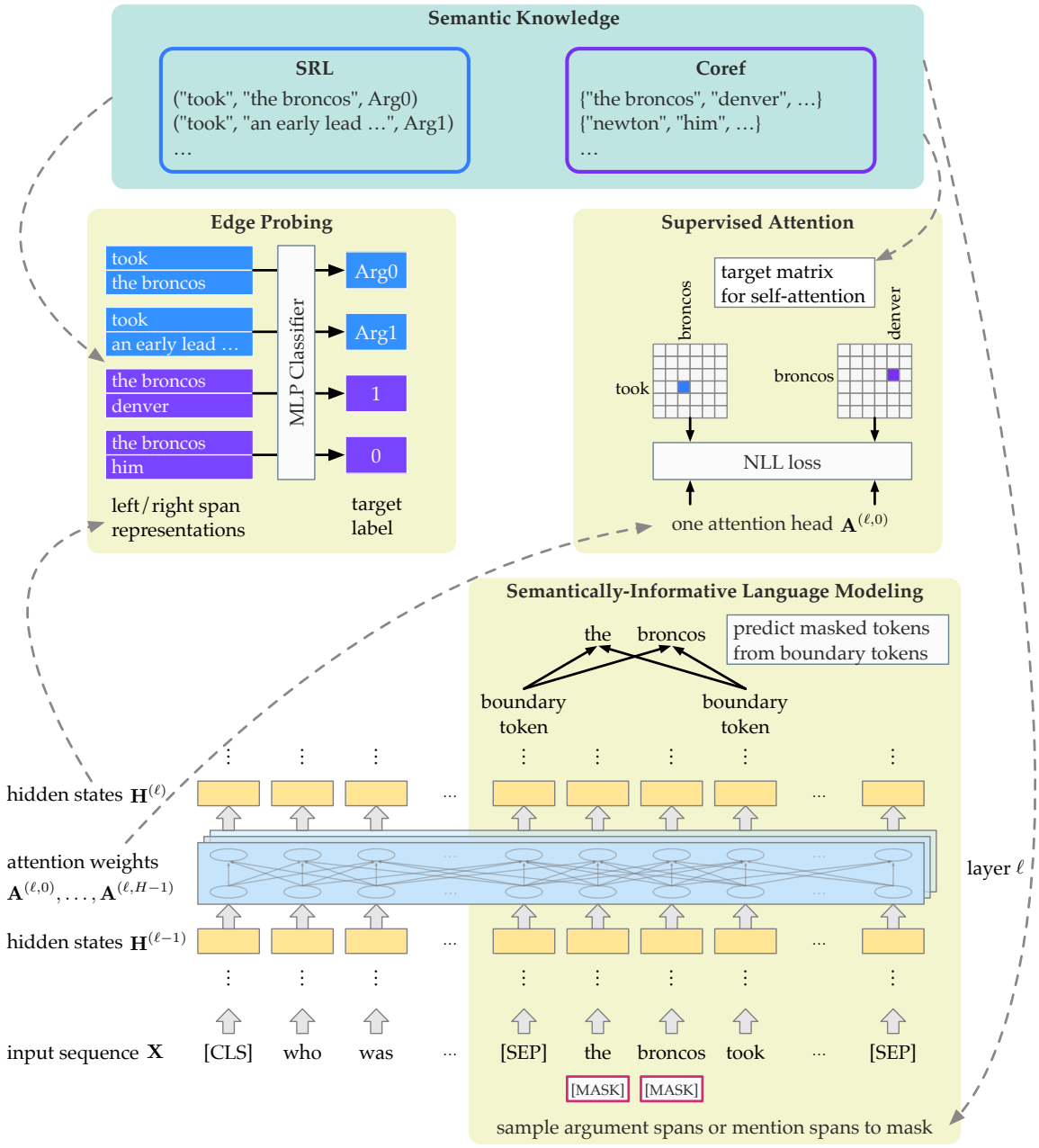


Figure 6.1: A graphical illustration of the three alternatives of integrating semantic knowledge.

our previous work in Chapter 5. We first construct a target 0/1 matrix for self-attention $\mathbf{A}^{target}$ from semantic knowledge, and the compute an additional loss term from the auxiliary supervision as:

$$\mathcal{L}_{SA} = \frac{1}{T}\sum_{i=1}^{T} -\log\left(\sum_{j=1}^{T} A_{i,j}^{(\ell,h)} \cdot A_{i,j}^{target}\right) \tag{6.7}$$

For SRL, the target matrix $\mathbf{A}^{target}$ is constructed by setting A_{ij}^{target} to 1 only if there exists a predicate-argument relation between x_i and x_j, i.e., either x_i is within an argument span of predicate x_j, or vice versa. For coreference, similarly, we set A_{ij}^{target} to 1 only if a mention containing x_i and a mention containing x_j are coreferent.

Semantically-Informative Language Modeling (LM) As we will find out in Section 6.5, swapping the base LM from BERT to some of its variants with different masking strategies in pre-training leads to different results, especially on out-of-domain datasets. For example, while BERT randomly samples sub-word tokens to mask, **BERT-whole-word-masking**[3] first samples full words and then masks all sub-word tokens within the sampled words. **SpanBERT** (Joshi et al., 2020) further augments that by sampling continuous spans of words and masking all sub-word tokens within the spans. We see that with everything else identical (including the model architecture and the number of parameters), BERT-whole-word-masking outperforms BERT, and SpanBERT further improves over BERT-whole-word-masking.

This leads us to hypothesize that a better masking strategy might be a key to higher generalization capabilities. Therefore, we try a third alternative to integrate semantic knowledge, that is, randomly sampling and masking some argument spans from the SRL tuples, or some mention spans from the coreference clusters, and then asking the model to recover masked spans via a language modeling objective. We consider the argument spans from SRL and the mention spans from coreference to be "semantically informative", as they usually hold essential information to the core meaning of the input text. Therefore, being able to recover these spans should principally indicate that the model understands semantics better.

[3] https://github.com/google-research/bert

For each sampled span (x_s, x_e) where (s, e) indicates its inclusive start and end positions, we use the "Span Boundary Objective" (SBO) from SpanBERT (Joshi et al., 2020) as an auxiliary loss term:

$$\mathcal{L}_{LM} = -\sum_{i=s}^{e} \log P(x_i | f(\mathbf{h}_{s-1}^{(\ell)}, \mathbf{h}_{e+1}^{(\ell)}, \mathbf{p}_{i-s+1})) \tag{6.8}$$

where function $f(\cdot)$ is a 2-layer MLP, $\mathbf{p}_{i-s+1}$ is a learned positional embedding to encode the relative position of x_i within the span, and ℓ is the index of the layer to get token embeddings from.

6.5 Experiments

We present some preliminary experimental results on the MRQA task. Since the test portions in Split II and Split III are not publicly available, we here focus on evaluating on the development portions in Split I and Split II, with the former being "in-domain" evaluation and the latter "out-of-domain" evaluation. In Table 6.2 we observe a strong correlation between performances on different portions of different splits. Therefore, we can faithfully assume that the scores on the development sets also indicate the model's performance on hidden test sets.

6.5.1 Implementation Details

Pre-processing As discussed in Subsection 6.4.2, we need to pre-process the data to get automatic annotations for SRL and coreference. Since SRL is labeled on a per-sentence basis, we first perform sentence segmentation on the passage text with the Pragmatic Segmenter tool[4]. For SRL prediction, we use an AllenNLP re-implementation of the BERT-based model from Shi and Lin (2019). For coreference resolution, we use the SpanBERT-based model[5] from Joshi et al. (2020).

Hyperparameters We largely follow the hyperparameter settings of the official baseline. We sample at most 75k examples from each of the 6 training datasets.

[4] https://github.com/diasks2/pragmatic_segmenter
[5] https://github.com/mandarjoshi90/coref

We use a batch size of 12 and train on 4 GPUs (thus an effective batch size of 48). We train the model for 2 epochs, using the Adam optimizer (Kingma and Ba, 2015) with weight decay, a learning rate of 3×10^{-5}, and a learning rate scheduler with linear warmup over the first 2,000 steps and linear decay afterwards. We implement our models using AllenNLP (Gardner et al., 2018) and the Transformers library from Hugging Face[6].

Base LMs The official baseline tests two variants of BERT: $\text{BERT}_{\text{BASE}}$, a 12-layer model with a hidden dimension of 768, and $\text{BERT}_{\text{LARGE}}$, a 24-layer model with a hidden dimension of 1024. In addition to that, we also experiment with two more variants of BERT as the base LM: $\text{BERT}_{\text{LARGE-WWM}}$ (whole word masking) and $\text{SpanBERT}_{\text{LARGE}}$. They both share the exactly same architecture with $\text{BERT}_{\text{LARGE}}$, but use different masking strategies for pre-training, as briefly discussed in Subsection 6.4.2. We use the pre-trained uncased models from the Transformers library for the former three, and the pre-trained cased model for SpanBERT[7].

Semantic Knowledge For the edge probing (EP) variant, we sample at most 30 SRL instances per example, or at most 15 positive instances and 15 negative instances when using coreference as the knowledge source. For the semantically-informative language modeling (LM) variant, we sample at most 10 spans to mask, with each span consisting of no more than 15 tokens. Also, the total number of masked tokens should not exceed 15% of all passage tokens. When training any of three variants, we use the hidden states or the attention weights of the top layer (i.e., set $\ell = 12$ for $*_{\text{BASE}}$ models and $\ell = 24$ for $*_{\text{LARGE}}$ models in Equation 6.6, 6.7, and 6.8) to compute the additional loss, which will be added to the original QA loss as defined in Equation 6.4.

6.5.2 Preliminary Results

We present some preliminary results in Table 6.3. Integrating semantic knowledge via edge probing (cells in light green) has almost no effect on in-domain performance, but gives a relatively consistent out-of-domain performance boost

[6] https://github.com/huggingface/transformers
[7] https://github.com/facebookresearch/SpanBERT

across the first 3 base LMs, though the scale of the improvement is quite marginal (usually below 1 point). The supervised attention approach (cells in light yellow), while usually leading to better in-domain performance, offers more varying improvement on out-of-domain data. The effect of the last alternative, semantically-informative language modeling (cells in light blue), is more mixed up, as sometimes it even leads to worse F_1 scores.

We also briefly try combining the semantic knowledge from SRL and Coref via the edge probing approach, as shown in cells in green. We do not observe consistent further improvement by the combination of two knowledge sources. In fact, in some cases we see even worse results than using a single knowledge source as supervision.

Another interesting observation is the comparison between $\text{BERT}_{\text{LARGE-WWM}}$ and $\text{SpanBERT}_{\text{LARGE}}$. While they achieve almost the same in-domain performance, there is a more than 2 points gap on out-of-domain performance. This supports our hypothesis that masking strategies might be critical to generalization, though the underlying reason is still unclear. Also, the EP approach gives a much larger improvement on $\text{BERT}_{\text{LARGE-WWM}}$ than on $\text{SpanBERT}_{\text{LARGE}}$. A possible explanation is that the span-based masking strategy in SpanBERT pre-training might already allow the model to learn representations that are more "span-aware" and better at predicting the edge labels.

Our best F_1 score on the out-of-domain Split II Dev sets, from integrating coreference knowledge as supervised attention on $\text{SpanBERT}_{\text{LARGE}}$, is already on par (69.37 vs. 69.7) with D-Net (Li et al., 2019), the best performing system in the shared task. However, since our goal is to examine whether semantic knowledge makes pre-trained LMs more generalizable to new domains, the numbers reported here are still disappointing, as we cannot see any consistent and significant improvement from any of the three alternatives of integrating semantic knowledge. Note that this does not conclude that semantic knowledge is not helpful, as we will discuss next some possible shortcomings of our current approaches.

6.5.3 Analysis

In Chapter 5, we find that coreference knowledge is particularly helpful on the LAMBADA task when applied as auxiliary supervision for self-attention on

Base LM	Semantic Knowledge		In-domain F_1 (Split I Dev)	Out-of-domain F_1 (Split II Dev)
$\text{BERT}_{\text{BASE}}$		–	76.27	56.73
	EP	SRL	76.21	57.41 (+0.68)
		Coref	76.34	57.28 (+0.55)
		SRL + Coref	77.31	56.76 (+0.03)
	SA	SRL	77.67	56.83 (+0.10)
		Coref	77.87	57.43 (+0.70)
	LM	SRL	77.27	56.87 (+0.14)
		Coref	77.32	56.58 (−0.15)
$\text{BERT}_{\text{LARGE}}$		–	79.51	61.60
	EP	SRL	79.40	62.04 (+0.44)
		Coref	79.61	62.26 (+0.66)
		SRL + Coref	80.81	62.52 (+0.92)
	SA	SRL	81.11	62.25 (+0.65)
		Coref	80.46	61.86 (+0.26)
	LM	SRL	80.40	61.54 (−0.06)
		Coref	79.41	60.77 (−0.83)
$\text{BERT}_{\text{LARGE-WWM}}$		–	82.23	65.89
	EP	SRL	81.89	66.94 (+1.05)
		Coref	82.02	66.46 (+0.57)
		SRL + Coref	82.99	66.86 (+0.97)
	SA	SRL	83.00	66.13 (+0.24)
		Coref	82.88	66.36 (+0.47)
	LM	SRL	82.66	65.74 (−0.15)
		Coref	82.62	64.28 (−1.61)
$\text{SpanBERT}_{\text{LARGE}}$		–	82.55	68.21
	EP	SRL	82.28	68.30 (+0.09)
		Coref	82.21	68.60 (+0.39)
		SRL + Coref	83.63	68.54 (+0.33)
	SA	SRL	83.45	68.89 (+0.68)
		Coref	83.13	69.37 (+1.16)
	LM	SRL	83.08	68.02 (−0.19)
		Coref	83.15	68.21 (+0.00)

Table 6.3: Evaluation results on MRQA by integrating semantic knowledge. The numbers in parentheses on the last column indicate the performance increase (+) or decrease (−) over the corresponding base LM without semantic knowledge.

a 4-layer transformer encoder. In this chapter, however, we do not get to the same conclusion. One possible explanation is that, these pre-trained LMs, containing hundreds of millions of parameters, are so powerful that they probably do not need all of the parameters to learn good representations for any downstream tasks. Therefore, whichever approach we choose to integrate semantic knowledge, as long as the knowledge is only used as an auxiliary multi-task learning objective, the model might be optimizing towards multiple objectives in a relatively independent manner. For example, in the supervised attention (SA) setting, we are only enforcing one attention head to attend to semantically-relevant part of the text, while the model could just rely on the remaining 11 (or 23) heads to do answer span prediction.

Also, the semantic knowledge as encoded in the edge probing task might just be too easy for pre-trained LMs to pick up. This is indicated by the observation that the additional loss term L_{EP} usually drops to a very small number (~0.2) compared to the main loss term for answer prediction at the end of training. Also, Tenney et al. (2019b) find that the pre-trained embeddings from BERT, even without any fine-tuning, are already doing very well at predicting SRL labels and coreference relations. This suggests that we might want some harder semantic task for the pre-trained LMs to actually learn from it.

We still believe that semantic knowledge can play an important role in making pre-trained LMs more robust and generalizable, and we discuss possible future directions in Section 7.1.

6.6 Chapter Summary

In this chapter, we present an ongoing study on how to integrate semantic knowledge, including SRL and coreference, into large-scale pre-trained language models. We propose three different methods for the integration: edge probing, supervised attention, or semantically-informative language modeling. Some preliminary experiments on the MRQA 2019 Shared Task do not show positive results, as we observe marginal and inconsistent improvement from injecting semantic knowledge. We further analyze possible shortcomings of our current approaches and provide some insights for future directions.

Chapter 7

Conclusion

Neural network models have been the main driving force behind the rapid progress of NLP research in the past decade. With massive training data and complex neural architectures, we have seen models approaching human performance on many natural language understanding benchmarks. However, the vulnerability of many models against adversarial attacks has raised questions about the extent to which these models truly understand human language, and whether they are merely memorizing statistical patterns. This raises the question to us of whether the traditional linguistic knowledge of semantics can still be beneficial in the neural era.

In this thesis, we investigated the idea of combining semantic knowledge with neural models. There has been a long line of prior work on learning from syntactic structure for downstream tasks, but very little work on exploiting semantic structure. We started with a certain type of semantic knowledge, the implicit predicate-argument relations, and studied how the knowledge of event and entity structures can help inferring implicit arguments. We then focused on integrating a wider variety of types of semantic knowledge into neural models in a more latent fashion.

Implicit argument prediction has been considered a very hard task, mainly due to the lack of human-annotated data. We proposed an argument cloze task to allow automatic creation synthetic data at scale (Chapter 3). The task is to recover a randomly removed event argument from a list of candidates, where event structures are extracted from dependency parsing, and all other arguments in the context are treated as candidates. With the argument cloze task, we created a large-scale training dataset from Wikipedia by running off-the-shelf dependency parsing and coreference resolution tools. We also created a synthetic evaluation dataset using the gold annotations of syntax and coreference in the OntoNotes corpus, which is larger and more diversified than existing human-annotated datasets.

With the synthetic training data, we developed two neural models for inferring implicit arguments. The first model, EVENTCOMP, draws on local narrative coherence, that is, whether a pair of events are likely to occur in the same context (Chap-

ter 3). We also found that entity salience features significantly boost performance. The second model, Pointer Attentive Reader, instead measures global narrative coherence, by casting the problem as reading comprehension (Chapter 4). We also introduced a multi-hop attention mechanism to tackle the harder cases with more than one implicit argument in a single event. Both these models perform well on a human-annotated nominal implicit arguments dataset based on NomBank, as well as the synthetic evaluation data from OntoNotes.

We also studied whether semantic knowledge can be integrated into neural models in a latent fashion, and thus benefiting downstream tasks. We proposed to inject semantic knowledge into a transformer encoder as the auxiliary supervision for self-attention (Chapter 5). On the LAMBADA word prediction task, which is designed to require long-distance reasoning, we demonstrated that a small model paired with coreference knowledge sets the new state-of-the-art, outperforming GPT-2, a much larger pre-trained language model.

Finally, we extended the idea of integrating semantic knowledge to the recent large-scale pre-trained LMs, like BERT and SpanBERT (Chapter 6). We explored the supervised self-attention approach which has been successful on in our previous work, along with two other alternatives of applying semantic knowledge: as edge probing instances for multi-task learning, or as semantically-informative masking for language modeling. On the MRQA 2019 Shared Task, we found that none of the three approaches can offer consistent and significant improvement across different base LMs. A possible explanation is that the approaches we have tried only provide a relatively weak signal that might not be enough to inject semantic knowledge into the highly-parameterized pre-trained LMs.

Overall, we showed that semantic knowledge can be utilized in many different ways to enhance neural models, and learn better representations. Even we faced some setback in our ongoing work of applying semantic knowledge to large-scale pre-trained LMs, we are still optimistic that semantic knowledge can be useful when integrated with some better approaches, which we will discuss in the next section. We look forward to continuing this exciting thread of research on designing more generalizable and robust models for language understanding.

7.1 Future Work

As with all areas of research, the work in this thesis answers some questions, but it also raise new ones. In this section, we discuss some possible future directions of research.

7.1.1 More Evaluation on Implicit Arguments

In Chapter 3 and Chapter 4, we mainly focus on evaluating our models for predicting implicit arguments on the G&C dataset (Gerber and Chai, 2010). While being one of the most widely used benchmark on implicit arguments, the dataset is still very small and limited, consisting of 1,247 examples covering just 10 nominal predicates. Therefore, performance on the G&C dataset might not fully reveal a model's strengths and weaknesses in reasoning implicit arguments.

The main reason for the sparsity of data is the difficulty in explicitly annotating implicit arguments, even for domain experts. Nonetheless, there has been some work on adapting resources from other domains to the task of implicit arguments. We describe some of them below. These datasets are considerable larger and more diverse than G&C. So it would be interesting to see how well our synthetic training data and proposed models can handle these different test cases.

Implied Predicate-Argument Relationships in RTE Stern and Dagan (2014) proposed a task of recognizing "Implied Predicate-Argument Relationships" in the context of textual inference, using the setting of *Recognizing Textual Entailment (RTE)*. The task of RTE (Dagan et al., 2005), is when given a pair of text fragments termed *Text* and *Hypothesis*, to recognize whether a human reading the Text would infer that the Hypothesis is most likely true. Here is a positive example:

(7.1)
Text: The crucial role Vioxx plays in Merck's portfolio was apparent last week when Merck's shares plunged 27 percent to 33 dollars after the withdrawal announcement.
Hypothesis: Merck withdrew Vioxx from the market.

To infer whether the entailment holds or not, one would typically need to match the lexical items and their relationships between the Text and the Hypothesis. In

this example, all important lexical items in the Hypothesis (e.g., "Merck", "withdrew", and "Vioxx") can be matched to corresponding words in the Text. However, while "Vioxx" is clearly the direct object of "withdrew" in the Hypothesis, there is no explicit syntactic relation between "withdrawal" and "Vioxx" in the Text. Therefore, if the entailment does hold, it is very likely that there is an **implied relationship** between "withdrawal" and "Vioxx".

Building on this observation, Stern and Dagan (2014) construct a dataset of implied predicate-argument relationships semi-automatically from the RTE-6 dataset Bentivogli et al. (2010). First, all explicit predicate-arguments relationships in the Text and the Hypothesis are automatically extracted using a dependency parser. Predicates and arguments in the Text are also aligned to the Hypothesis with some simple heuristics. Then, for every predicate-argument relationship that is explicitly stated in the Hypothesis but not in the Text, a human reader is asked to annotate whether the implied relationship indeed holds in the Text. The dataset consists of 4,022 instances, where 56% of them are annotated as positive instances.

Multi-sentence AMR O'Gorman et al. (2018) annotate a multi-sentence AMR (**MS-AMR**) corpus on top of exiting gold AMR annotations, extending them with coreference relations and implicit semantic roles cross sentence boundaries. In Example (7.2), annotators link **coreferent variables** (e.g., ARG0 of the predicate "leave-11" and ARG1 of the predicate "arrive-01" both refer to the person "Bill", as marked in red), as well as **fillers for implicit roles** (e.g., ARG2 of "leave-11" is also the implicit ARG4 of "arrive-01", as marked in blue).

(7.2)

Bill left for Paris.
(l / leave-11
:ARG0 (p / person :wiki - :name (n / name :op1 "Bill"))
:ARG2 (c / city :wiki "Paris" :name (n / name :op1 "Paris")))
He arrived at noon.
(a / arrive-01
:ARG1 (h / he)
:ARG3 (*i / implicit role: start point*)
:ARG4 (*i2 / implicit role: end point; destination*)
:TIME (d / date-entity :dayperiod (n3 / noon)))

O'Gorman et al. (2018) annotate 293 documents, covering roughly 10% of the total AMR corpus. The dataset contains ~8,000 predicates, with ~2,400 implicit roles identified. The MS-AMR annotation is included as part of the most recent AMR public release[1].

RAMS Ebner et al. (2020) construct a **R**oles **A**cross **M**ultiple **S**entences dataset, by applying the AIDA-1 ontology[2] (Phase 1 of the DARPA AIDA project, developed for recent geopolitical events related to Russia and Ukraine) to annotate news articles. The authors first collect a set of topically relevant news articles, and manually map each event ((sub-)sub)type in the ontology to a list of high-precision event triggers (lexical units likely to evoke the event, for example, *arson* is a trigger for the `Conflict.Attack.SetFire` event). The Crowd workers are then asked to label the closes argument span for each role of each matched event trigger in a 5-sentence context window (two sentences before the sentence with the trigger and two sentences after).

The dataset contains over 9,000 events and 21,000 arguments, covering all 139 event types and 65 role types in the AIDA-1 ontology. About 18% of the labeled arguments are **implicit arguments**, occurring outside the sentence with the event trigger. In the following example, the word "bombarding" in the second sentence triggers an event of type `Conflict.Attack.AirstrikeMissileAttack`. The event is labeled with four arguments, two in the same sentence: "Russian" as the `attacker`, "rebel post" as the `target`, and two in the previous sentence: "aircraft" as the `instrument`, "Syria" as the `place`.

(7.3) When Russian aircraft bombed a remote garrison in southeastern Syria last month, alarm bells sounded at the Pentagon and the Ministry of Defense in London.
The Russians weren't ***bombarding*** a run-of-the-mill rebel outpost, according to U.S. officials.

[1] LDC Catalog No. LDC2020T02
[2] https://tac.nist.gov/2019/SM-KBP/data.html

7.1.2 New Methods for Integrating Semantic Knowledge

In Chapter 5, we find that integrating coreference knowledge into a relatively small-scale model significantly improves performance on the LAMBADA task. In Chapter 6, however, the same technique and two other alternatives of integrating semantic knowledge do not help as much when applied on large-scale pre-trained models. We analyze some possible reasons of the setback in Subsection 6.5.3, here we discuss future directions for new methods to address them.

Stronger Supervision from Auxiliary Tasks The edge probing approach applies semantic knowledge as a simple pair-wise classification problem. For SRL, it predicts the semantic role label between a predicate and an argument. For coreference, it predicts whether two mentions are coreferent or not. This setting, however, does not fully exploit the complexity of SRL and coreference structures, by assuming that the gold spans and the predicate-argument alignments are all known as input. As a result, the auxiliary loss value drops quickly during training, indicating that the model might not be actually learning semantic knowledge from it.

One possible fix is for the model to do full-scale structural predictions of SRL and coreference via multi-task learning. For example, taking the hidden states from one transformer layer, the model needs to first identify all predicates and then perform argument identification and classification for each predicate as a sequence labeling task, similar to the framework in recent end-to-end SRL systems (He et al., 2018, Shi and Lin, 2019). Or in the context of coreference knowledge, instead of doing binary prediction of positive / negative mention pairs, the model first computes all span representations and then jointly predicts the antecedent of every mention, also similar to recent coreference resolution models (Lee et al., 2017; 2018). In principle, these structural prediction tasks are much harder than the edge probing tasks we have already experimented, and would hopefully provide stronger supervision signals for the model to better learn semantic knowledge.

Explicitly Modeling Events and Entities So far, our efforts in integrating semantic knowledge have been on applying it through multi-task learning, without changing the underlying model architecture. This implicit way of integration could be less ideal, in that large-scale pre-trained LMs might have excessive ex-

pressing power to learn different tasks independently without effectively posing inductive biases on the end task. Therefore, we want to see if it is possible to integrate semantic knowledge in a more explicit manner. As a quick reminder, SRL models the predicate-argument relations in events, and coreference resolution models coreferent mentions of entities. We can thus try to first dynamically compute representations of events and entities in the text using the knowledge of SRL and coreference as an intermediate , and then integrate these event and entity representations back to the model to update token-level representation.

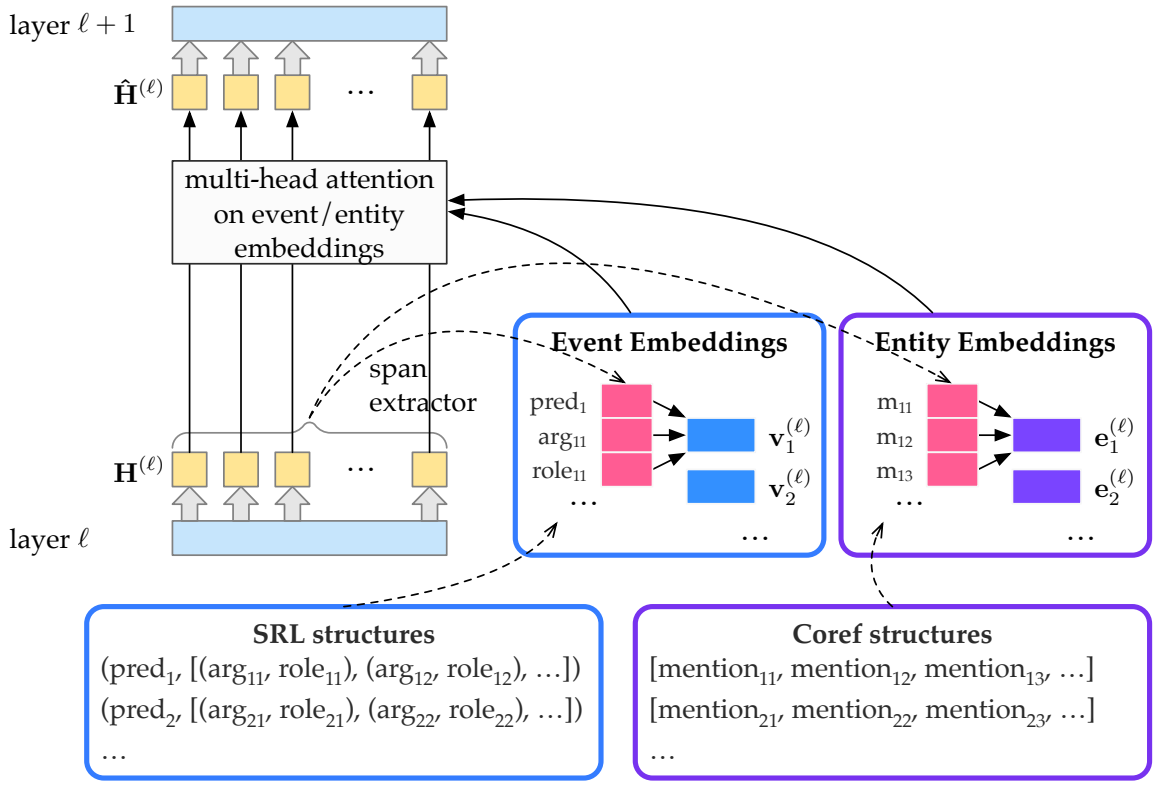


Figure 7.1: A diagram of the proposed method to explicitly model events and entities using semantic knowledge.

To illustrate, given some annotations of SRL and coreference, from the hidden states $\mathbf{H}^{(\ell)}$ after layer ℓ, we construct a list of event embeddings $\{\mathbf{v}_1^{(\ell)}, \ldots, \mathbf{v}_M^{(\ell)}\}$ by aggregating predicate span representations and argument span representations, and a list of entity embeddings $\{\mathbf{e}_1^{(\ell)}, \ldots, \mathbf{e}_N^{(\ell)}\}$ by aggregating the mention span representations in each coreference chain. Then we apply a multi-head attention layer between $\mathbf{H}^{(\ell)}$ and the event / entity embeddings to get enhanced token-level

representations $\hat{\mathbf{H}}^{(\ell)}$, which are fed into the next layer $\ell + 1$ as input. Figure 7.1 provides a graphical illustration.

The downside of the approach is that we would also need SRL and coreference information at test time. This might be addressed by combining this approach with the previous future direction we just discussed, i.e., at training time, we can also take the hidden states of another transformer layer j ($j < \ell$) to do a multi-task learning on the structural prediction of SRL and coreference. Then at test time, we can derive events and entities from the predicted SRL and coreference labels.

The idea is inspired by several lines of previous work. The most relevant recent work is KnowBERT (Peters et al., 2019a), in which a KAR (Knowledge Attention and Recontextualization) mechanism in proposed to integrate background knowledge of world entities from external KBs into the pre-training of BERT. This is also related to the work by Ji et al. (2017) on enhancing a neural language model by explicitly modeling entities. The proposed EntityNLM model keeps track of whether each word is part of an entity mention and which entity it refers to via a set of latent variable, and dynamically updates entity representations which will be used to predict the next word during the generation process. Also, the idea can be linked to the previous line of work on memory networks (Weston et al., 2015, Sukhbaatar et al., 2015), as it can be viewed as storing the information about entities and events in a set of memory slots.

Bibliography

Sungjin Ahn, Heeyoul Choi, Tanel Pärnamaa, and Yoshua Bengio. A neural knowledge language model. *arXiv preprint arXiv:1608.00318*, 2016.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *Proc. Linguistic Coreference Workshop at the first Conf. on Language Resources and Evaluation (LREC), Granada, Spain, May 1998*, volume 1, pages 563–566, 1998.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1409.0473.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal, Quebec, Canada, August 1998. Association for Computational Linguistics. doi:10.3115/980845.980860. URL https://www.aclweb.org/anthology/P98-1013.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W13-2322.

Srinivas Bangalore and Aravind K. Joshi. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–265, 1999. URL https://www.aclweb.org/anthology/J99-2004.

Lisa Bauer, Yicheng Wang, and Mohit Bansal. Commonsense for generative multi-hop question answering tasks. In *Proceedings of the 2018 Conference*

on Empirical Methods in Natural Language Processing, pages 4220–4230, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi:10.18653/v1/D18-1454. URL https://www.aclweb.org/anthology/D18-1454.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The sixth PASCAL recognizing textual entailment challenge. In *Proceedings of the Third Text Analysis Conference, TAC 2010, Gaithersburg, Maryland, USA, November 15-16, 2010*. NIST, 2010. URL https://tac.nist.gov/publications/2010/additional.papers/RTE6_overview.proceedings.pdf.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi:10.18653/v1/D15-1075. URL https://www.aclweb.org/anthology/D15-1075.

Rui Cai and Mirella Lapata. Syntax-aware semantic role labeling without parsing. *Transactions of the Association for Computational Linguistics*, 7:343–356, March 2019. doi:10.1162/tacl_a_00272. URL https://www.aclweb.org/anthology/Q19-1022.

Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P13-2131.

Xavier Carreras and Lluís Màrquez. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W05-0620.

Nathanael Chambers and Dan Jurafsky. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P08-1090.

Nathanael Chambers and Dan Jurafsky. Unsupervised learning of narrative schemas and their participants. 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 602–610, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P09-1068.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1082. URL https://www.aclweb.org/anthology/D14-1082.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 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, August 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1223. URL https://www.aclweb.org/anthology/P16-1223.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Diana Inkpen, and Si Wei. Neural natural language inference models enhanced with external knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2406–2417, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi:10.18653/v1/P18-1224. URL https://www.aclweb.org/anthology/P18-1224.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas, November 2016. Association for Computational Linguistics. doi:10.18653/v1/D16-1053. URL https://www.aclweb.org/anthology/D16-1053.

Pengxiang Cheng and Katrin Erk. Implicit argument prediction with event knowledge. 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 831–840, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/N18-1076. URL https://www.aclweb.org/anthology/N18-1076.

Pengxiang Cheng and Katrin Erk. Implicit argument prediction as reading comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6284–6291, 2019.

Pengxiang Cheng and Katrin Erk. Attending to entities for better text understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.

Christian Chiarcos and Niko Schenk. Memory-based acquisition of argument structures and its application to implicit role detection. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 178–187, Prague, Czech Republic, September 2015. Association for Computational Linguistics. doi:10.18653/v1/W15-4626. URL https://www.aclweb.org/anthology/W15-4626.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California, June 2016. Association for Computational Linguistics. doi:10.18653/v1/N16-1012. URL https://www.aclweb.org/anthology/N16-1012.

Zewei Chu, Hai Wang, Kevin Gimpel, and David McAllester. Broad context language modeling as reading comprehension. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 52–57, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-2009.

Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi:10.18653/v1/P18-1078. URL https://www.aclweb.org/anthology/P18-1078.

Kevin Clark and Christopher D. Manning. Entity-centric coreference resolution with model stacking. 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)*, pages 1405–1415, Beijing, China, July 2015. Association for Computational Linguistics. doi:10.3115/v1/P15-1136. URL https://www.aclweb.org/anthology/P15-1136.

Kevin Clark and Christopher D. Manning. 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, Berlin, Germany, August 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1061. URL https://www.aclweb.org/anthology/P16-1061.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer, 2005.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087, 2015.

Zeyu Dai and Ruihong Huang. A regularization approach for incorporating event knowledge and coreference relations into neural discourse parsing. 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 2976–2987, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1295. URL https://www.aclweb.org/anthology/D19-1295.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. 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 5925–5932, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1606. URL https://www.aclweb.org/anthology/D19-1606.

Donald Davidson. The logical form of action sentences. *The logic of decision and action*, pages 81–95, 1967.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1807.03819.

Pascal Denis and Jason Baldridge. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 660–669, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D08-1069.

Pascal Denis and Jason Baldridge. Global joint models for coreference resolution and named entity classification. *Procesamiento del Lenguaje Natural*, 42, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi:10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi:10.18653/v1/P17-1168. URL https://www.aclweb.org/anthology/P17-1168.

Bhuwan Dhingra, Qiao Jin, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. Neural models for reasoning over multiple mentions using coreference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 42–48, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/N18-2007. URL https://www.aclweb.org/anthology/N18-2007.

Quynh Ngoc Thi Do, Steven Bethard, and Marie-Francine Moens. Improving implicit semantic role labeling by predicting semantic frame arguments. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 90–99, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. URL https://www.aclweb.org/anthology/I17-1010.

David Dowty. Thematic proto-roles and argument selection. *Language*, 67(3):547–619, 1991.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378,

Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi:10.18653/v1/N19-1246. URL https://www.aclweb.org/anthology/N19-1246.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

Jesse Dunietz and Daniel Gillick. A new entity salience task with millions of training examples. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 205–209, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. doi:10.3115/v1/E14-4040. URL https://www.aclweb.org/anthology/E14-4040.

Greg Durrett and Dan Klein. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971–1982, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1203.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. Multi-sentence argument linking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020. URL https://arxiv.org/abs/1911.03766.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.

Parvin Sadat Feizabadi and Sebastian Padó. Combining seemingly incompatible corpora for implicit semantic role labeling. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 40–50, Denver, Colorado, June 2015. Association for Computational Linguistics. doi:10.18653/v1/S15-1005. URL https://www.aclweb.org/anthology/S15-1005.

Charles Fillmore. The case for case. *Universals in Linguistic Theory*, 1968.

John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-5801. URL https://www.aclweb.org/anthology/D19-5801.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A. Smith. A discriminative graph-based parser for the Abstract Meaning Representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi:10.3115/v1/P14-1134. URL https://www.aclweb.org/anthology/P14-1134.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi:10.18653/v1/W18-2501. URL https://www.aclweb.org/anthology/W18-2501.

Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, et al. Evaluating NLP models via contrast sets. *arXiv preprint arXiv:2004.02709*, 2020.

Matthew Gerber and Joyce Chai. Beyond NomBank: A study of implicit arguments for nominal predicates. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1583–1592, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P10-1160.

Matthew Gerber and Joyce Y. Chai. Semantic role labeling of implicit arguments for nominal predicates. *Computational Linguistics*, 38(4):755–798, 2012. doi:10.1162/COLI_a_00110. URL https://www.aclweb.org/anthology/J12-4003.

Mor Geva, Yoav Goldberg, and Jonathan Berant. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. 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 1161–1166, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1107. URL https://www.aclweb.org/anthology/D19-1107.

Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 512–520, Hong Kong, October 2000. Association for Computational Linguistics. doi:10.3115/1075218.1075283. URL https://www.aclweb.org/anthology/P00-1065.

Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002. doi:10.1162/089120102760275983. URL https://www.aclweb.org/anthology/J02-3001.

Philip Gorinski, Josef Ruppenhofer, and Caroline Sporleder. Towards weakly supervised resolution of null instantiations. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 119–130, Potsdam, Germany, March 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W13-0111.

Mark Granroth-Wilding and Stephen Clark. What happens next? event prediction using a compositional neural network model. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 107–112, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/N18-2017. URL https://www.aclweb.org/anthology/N18-2017.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 1–18, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W09-1201.

Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. Deep semantic role labeling: What works and what's next. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 473–483, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi:10.18653/v1/P17-1044. URL https://www.aclweb.org/anthology/P17-1044.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. Jointly predicting predicates and arguments in neural semantic role labeling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 364–369, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi:10.18653/v1/P18-2058. URL https://www.aclweb.org/anthology/P18-2058.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701, 2015.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The Goldilocks Principle: Reading children's books with explicit memory representations. In *International Conference on Learning Representations*, 2016. URL http://arxiv.org/abs/1511.02301.

Luong Hoang, Sam Wiseman, and Alexander Rush. Entity tracking improves cloze-style reading comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1055, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi:10.18653/v1/D18-1130. URL https://www.aclweb.org/anthology/D18-1130.

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

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA, June 2006. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N06-2015.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1356. URL https://www.aclweb.org/anthology/P19-1356.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1830–1839, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi:10.18653/v1/D17-1195. URL https://www.aclweb.org/anthology/D17-1195.

Robin Jia and Percy Liang. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany, August 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1002. URL https://www.aclweb.org/anthology/P16-1002.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Nat-*

ural Language Processing, pages 2021–2031, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi:10.18653/v1/D17-1215. URL https://www.aclweb.org/anthology/D17-1215.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi:10.18653/v1/P17-1147. URL https://www.aclweb.org/anthology/P17-1147.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. BERT for coreference resolution: Baselines and analysis. 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 5803–5808, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1588. URL https://www.aclweb.org/anthology/D19-1588.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 908–918, Berlin, Germany, August 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1086. URL https://www.aclweb.org/anthology/P16-1086.

Jungo Kasai, Dan Friedman, Robert Frank, Dragomir Radev, and Owen Rambow. Syntax-aware neural semantic role labeling with supertags. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 701–709, Minneapolis, Minnesota, June 2019. Association for Computa-

tional Linguistics. doi:10.18653/v1/N19-1075. URL https://www.aclweb.org/anthology/N19-1075.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1181. URL https://www.aclweb.org/anthology/D14-1181.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1609.02907.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018. doi:10.1162/tacl_a_00023. URL https://www.aclweb.org/anthology/Q18-1023.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi:10.18653/v1/D17-1082. URL https://www.aclweb.org/anthology/D17-1082.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 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, June 2016. Association for Computational Linguistics. doi:10.18653/v1/N16-1030. URL https://www.aclweb.org/anthology/N16-1030.

Egoitz Laparra and German Rigau. ImpAr: A deterministic algorithm for implicit semantic role labelling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1180–1189, Sofia, Bulgaria, August 2013a. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P13-1116.

Egoitz Laparra and German Rigau. Sources of evidence for implicit argument resolution. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 155–166, Potsdam, Germany, March 2013b. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W13-0114.

Anne Lauscher, Ivan Vulić, Edoardo Maria Ponti, Anna Korhonen, and Goran Glavaš. Specializing unsupervised pretraining models for word-level semantic similarity. *arXiv preprint arXiv:1909.02339*, 2019.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916, 2013. doi:10.1162/COLI_a_00152. URL https://www.aclweb.org/anthology/J13-4004.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi:10.18653/v1/D17-1018. URL https://www.aclweb.org/anthology/D17-1018.

Kenton Lee, Luheng He, and Luke Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/N18-2108. URL https://www.aclweb.org/anthology/N18-2108.

Seanie Lee, Donggyu Kim, and Jangwon Park. Domain-agnostic question-answering with adversarial training. In *Proceedings of the 2nd Workshop on Ma-*

chine Reading for Question Answering, pages 196–202, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-5826. URL https://www.aclweb.org/anthology/D19-5826.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Hongyu Li, Xiyuan Zhang, Yibing Liu, Yiming Zhang, Quan Wang, Xiangyang Zhou, Jing Liu, Hua Wu, and Haifeng Wang. D-NET: A pre-training and fine-tuning framework for improving the generalization of machine reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 212–219, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-5828. URL https://www.aclweb.org/anthology/D19-5828.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. KagNet: Knowledge-aware graph networks for commonsense reasoning. 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 2829–2839, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1282. URL https://www.aclweb.org/anthology/D19-1282.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. Barack's wife Hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5962–5971, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1598. URL https://www.aclweb.org/anthology/P19-1598.

Shayne Longpre, Yi Lu, Zhucheng Tu, and Chris DuBois. An exploration of data augmentation and sampling techniques for domain-agnostic question answering. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 220–227, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-5829. URL https://www.aclweb.org/anthology/D19-5829.

Xiaoqiang Luo. 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, October 2005. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/H05-1004.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi:10.18653/v1/D15-1166. URL https://www.aclweb.org/anthology/D15-1166.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi:10.3115/v1/P14-5010. URL https://www.aclweb.org/anthology/P14-5010.

Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi:10.18653/v1/D17-1159. URL https://www.aclweb.org/anthology/D17-1159.

Diego Marcheggiani and Ivan Titov. Graph convolutions over constituent trees for syntax-aware semantic role labeling. *arXiv preprint arXiv:1909.09814*, 2019.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi:10.18653/v1/K17-1041. URL https://www.aclweb.org/anthology/K17-1041.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL https://www.aclweb.org/anthology/J93-2004.

Christian Matthiessen and John A Bateman. *Text generation and systemic-functional linguistics: experiences from English and Japanese*. Pinter Publishers, 1991.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305, 2017.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. The NomBank project: An interim report. In *Proceedings of the Workshop Frontiers in Corpus Annotation at HLT-NAACL 2004*, pages 24–31, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W04-2705.

Julian Michael, Gabriel Stanovsky, Luheng He, Ido Dagan, and Luke Zettlemoyer. Crowdsourcing question-answer meaning representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 560–568, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/N18-2089. URL https://www.aclweb.org/anthology/N18-2089.

Todor Mihaylov and Anette Frank. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832, Melbourne, Australia, July 2018.

Association for Computational Linguistics. doi:10.18653/v1/P18-1076. URL https://www.aclweb.org/anthology/P18-1076.

Todor Mihaylov and Anette Frank. Discourse-aware semantic self-attention for narrative reading comprehension. 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 2541–2552, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1257. URL https://www.aclweb.org/anthology/D19-1257.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapa Nakashole, Emmanouil Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard Wang, Derry Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. Never-ending learning. In *AAAI Conference on Artificial Intelligence*, 2015. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/10049.

Timothy Niven and Hung-Yu Kao. Probing neural network comprehension of natural language arguments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4658–4664, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1459. URL https://www.aclweb.org/anthology/P19-1459.

Tim O'Gorman, Michael Regan, Kira Griffitt, Ulf Hermjakob, Kevin Knight, and Martha Palmer. AMR beyond the sentence: the multi-sentence AMR corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3693–3702, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C18-1313.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005. doi:10.1162/0891201053630264. URL https://www.aclweb.org/anthology/J05-1004.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1144. URL https://www.aclweb.org/anthology/P16-1144.

Terence Parsons. *Events in the Semantics of English*, volume 334. MIT press Cambridge, MA, 1990.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 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, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi:10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge enhanced contextual word 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)*, pages 43–54, Hong Kong, China, November 2019a. Association for Computational Linguistics. doi:10.18653/v1/D19-1005. URL https://www.aclweb.org/anthology/D19-1005.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 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, August 2019b. Association for Computational Linguistics. doi:10.18653/v1/W19-4302. URL https://www.aclweb.org/anthology/W19-4302.

Karl Pichotta and Raymond Mooney. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–229, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. doi:10.3115/v1/E14-1024. URL https://www.aclweb.org/anthology/E14-1024.

Karl Pichotta and Raymond J Mooney. Learning statistical scripts with LSTM recurrent neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016a.

Karl Pichotta and Raymond J. Mooney. Using sentence-level LSTM language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 279–289, Berlin, Germany, August 2016b. Association for Computational Linguistics. doi:10.18653/v1/P16-1027. URL https://www.aclweb.org/anthology/P16-1027.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany, August 2016.

Association for Computational Linguistics. doi:10.18653/v1/P16-2067. URL https://www.aclweb.org/anthology/P16-2067.

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. Semantic role labeling using different syntactic views. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 581–588, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi:10.3115/1219840.1219912. URL https://www.aclweb.org/anthology/P05-1072.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Joint Conference on EMNLP and CoNLL - Shared Task*, pages 1–40, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W12-4501.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W13-3516.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2): 257–287, 2008. doi:10.1162/coli.2008.34.2.257. URL https://www.aclweb.org/anthology/J08-2005.

Delai Qiu, Yuanzhe Zhang, Xinwei Feng, Xiangwen Liao, Wenbin Jiang, Yajuan Lyu, Kang Liu, and Jun Zhao. Machine reading comprehension using structural knowledge graph-aware network. 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 5896–5901, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1602. URL https://www.aclweb.org/anthology/D19-1602.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. 2018. URL https://openai.com/blog/language-unsupervised/.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL https://openai.com/blog/better-language-models/.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Altaf Rahman and Vincent Ng. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 968–977, Singapore, August 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D09-1101.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 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, November 2016. Association for Computational Linguistics. doi:10.18653/v1/D16-1264. URL https://www.aclweb.org/anthology/D16-1264.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi:10.18653/v1/P18-2124. URL https://www.aclweb.org/anthology/P18-2124.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1020.

Michael Roth and Anette Frank. Inducing implicit arguments from comparable texts: A framework and its applications. *Computational Linguistics*, 41(4):625–664, December 2015. doi:10.1162/COLI_a_00236. URL https://www.aclweb.org/anthology/J15-4003.

Michael Roth and Mirella Lapata. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202, Berlin, Germany, August 2016. Association for Computational Linguistics. doi:10.18653/v1/P16-1113. URL https://www.aclweb.org/anthology/P16-1113.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1681–1686, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi:10.18653/v1/D15-1195. URL https://www.aclweb.org/anthology/D15-1195.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. SemEval-2010 task 10: Linking events and their participants in discourse. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 45–50, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S10-1008.

Roger C Schank and Robert Abelson. Scripts, goals, plans, and understanding, 1977.

Niko Schenk and Christian Chiarcos. Unsupervised learning of prototypical fillers for implicit semantic role labeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1473–1479, San Diego, California, June 2016.

Association for Computational Linguistics. doi:10.18653/v1/N16-1173. URL https://www.aclweb.org/anthology/N16-1173.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*, 2017. URL http://arxiv.org/abs/1611.01603.

Peng Shi and Jimmy Lin. Simple BERT models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*, 2019.

Carina Silberer and Anette Frank. Casting implicit role linking as an anaphora resolution task. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 1–10, Montréal, Canada, 7-8 June 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S12-1001.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27 (4):521–544, 2001. doi:10.1162/089120101753342653. URL https://www.aclweb.org/anthology/J01-4004.

Robyn Speer and Catherine Havasi. Representing general relational knowledge in ConceptNet 5. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3679–3686, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/1072_Paper.pdf.

Robyn Speer, Joshua Chin, and Catherine Havasi. ConceptNet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Asher Stern and Ido Dagan. Recognizing implied predicate-argument relationships in textual inference. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 739–744, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi:10.3115/v1/P14-2120. URL https://www.aclweb.org/anthology/P14-2120.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5027–5038, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi:10.18653/v1/D18-1548. URL https://www.aclweb.org/anthology/D18-1548.

Sainbayar Sukhbaatar, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. LSTM neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Swabha Swayamdipta, Sam Thomson, Kenton Lee, Luke Zettlemoyer, Chris Dyer, and Noah A. Smith. Syntactic scaffolds for semantic structures. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3772–3782, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi:10.18653/v1/D18-1412. URL https://www.aclweb.org/anthology/D18-1412.

Takumi Takahashi, Motoki Taniguchi, Tomoki Taniguchi, and Tomoko Ohkuma. CLER: Cross-task learning with expert representation to generalize reading and understanding. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 183–190, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-5824. URL https://www.aclweb.org/anthology/D19-5824.

Alon Talmor and Jonathan Berant. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the*

57th Annual Meeting of the Association for Computational Linguistics, pages 4911–4921, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1485. URL https://www.aclweb.org/anthology/P19-1485.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi:10.18653/v1/N19-1421. URL https://www.aclweb.org/anthology/N19-1421.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy, July 2019a. Association for Computational Linguistics. doi:10.18653/v1/P19-1452. URL https://www.aclweb.org/anthology/P19-1452.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*, 2019b. URL https://arxiv.org/abs/1905.06316.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*, pages 45–52. Association for Computational Linguistics, 1995.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700, 2015.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing NLP. 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 2153–2162, Hong Kong, China, November 2019. Association for Computational Linguistics. doi:10.18653/v1/D19-1221. URL https://www.aclweb.org/anthology/D19-1221.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3261–3275, 2019a.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019b. URL https://arxiv.org/abs/1804.07461.

Chao Wang and Hui Jiang. Explicit utilization of general knowledge in machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2263–2272, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1219. URL https://www.aclweb.org/anthology/P19-1219.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. A transition-based algorithm for AMR parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–375, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi:10.3115/v1/N15-1040. URL https://www.aclweb.org/anthology/N15-1040.

Shuohang Wang and Jing Jiang. Machine comprehension using match-LSTM and answer pointer. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1608.07905.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi:10.18653/v1/P17-1018. URL https://www.aclweb.org/anthology/P17-1018.

Dirk Weissenborn, Tomáš Kočiskỳ, and Chris Dyer. Dynamic integration of background knowledge in neural nlu systems. *arXiv preprint arXiv:1706.02596*, 2017.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302, 2018. doi:10.1162/tacl_a_00021. URL https://www.aclweb.org/anthology/Q18-1021.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1410.3916.

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

Wei Wu, Fei Wang, Arianna Yuan, Fei Wu, and Jiwei Li. Coreference resolution as query-based span prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020. URL https://arxiv.org/abs/1911.01746.

Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1611.01604.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJlzm64tDH.

An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. Enhancing pre-trained language representations with rich knowledge

for machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2346–2357, Florence, Italy, July 2019a. Association for Computational Linguistics. doi:10.18653/v1/P19-1226. URL https://www.aclweb.org/anthology/P19-1226.

Bishan Yang and Tom Mitchell. Leveraging knowledge bases in LSTMs for improving machine reading. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1436–1446, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi:10.18653/v1/P17-1132. URL https://www.aclweb.org/anthology/P17-1132.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi:10.18653/v1/D18-1259. URL https://www.aclweb.org/anthology/D18-1259.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019b.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1850–1859, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi:10.18653/v1/D17-1197. URL https://www.aclweb.org/anthology/D17-1197.

Ziqing Yang, Yiming Cui, Wanxiang Che, Ting Liu, Shijin Wang, and Guoping Hu. Improving machine reading comprehension via adversarial training. *arXiv preprint arXiv:1911.03614*, 2019c.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. Fast and accurate reading comprehension by combining self-attention and con-

volution. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B14TlG-RW.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi:10.18653/v1/D18-1009. URL https://www.aclweb.org/anthology/D18-1009.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. ReCoRD: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*, 2018.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. AMR parsing as sequence-to-graph transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 80–94, Florence, Italy, July 2019a. Association for Computational Linguistics. doi:10.18653/v1/P19-1009. URL https://www.aclweb.org/anthology/P19-1009.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy, July 2019b. Association for Computational Linguistics. doi:10.18653/v1/P19-1139. URL https://www.aclweb.org/anthology/P19-1139.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. Semantics-aware BERT for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.

Jie Zhou and Wei Xu. End-to-end learning of semantic role labeling using recurrent 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 (Volume 1: Long Papers)*, pages 1127–1137, Beijing, China, July 2015. Association for Computational Linguistics. doi:10.3115/v1/P15-1109. URL https://www.aclweb.org/anthology/P15-1109.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.

Vita

Pengxiang Cheng was born in 1992 in Huangshan, China. He graduated from Tsinghua University in 2013, with a Bachelor of Engineering in Automation, and a second Bachelor's degree in Economics. After that, he entered the Graduate School at the University of Texas at Austin for his doctoral studies in Computer Science, where he has been working on natural language understanding.

Address: pengxiang.cheng@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.