Implicit Argument Prediction as Reading Comprehension

Pengxiang Cheng
Department of Computer Science
The University of Texas at Austin
pxcheng@utexas.edu

Katrin Erk
Department of Linguistics
The University of Texas at Austin
katrin.erk@mail.utexas.edu

Abstract

Implicit arguments, which cannot be detected solely through syntactic cues, make it harder to extract predicate-argument tuples. We present a new model for implicit argument prediction that draws on reading comprehension, casting the predicate-argument tuple with the missing argument as a query. We also draw on pointer networks and multi-hop computation. Our model shows good performance on an argument cloze task as well as on a nominal implicit argument prediction task.

1 Introduction

Predicate-argument tuples describe "who did what to whom" and are an important data structure to extract from text, for example in Open Information Extraction (Etzioni et al. 2007). 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, for example:[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, *Gillingham* is an implicit argument to *decline*. Generally, predicates with implicit arguments can be nouns, as in the example, or verbs.

Implicit argument prediction as a machine learning task was introduced by Gerber and Chai (2010) and Ruppenhofer et al. (2010), and was studied in a number of papers (Silberer and Frank 2012; Laparra and Rigau 2013a; Stern and Dagan 2014; Chiarcos and Schenk 2015; Schenk and Chiarcos 2016; Do, Bethard, and Moens 2017). In this task, the model is given a predicate-argument tuple with one or more arguments missing. The model then chooses a filler for each missing argument from the document (or chooses to leave the argument unfilled). Building on recent work that made the task accessible to neural models through training on automatically generated data (Cheng and Erk 2018), we introduce a new neural model for implicit argument prediction.

In this paper, we view the task of implicit argument prediction as related to Reading Comprehension (Hermann et al. 2015): A predicate-argument tuple with the missing argument is a query. The answer to the query has to be located in the document. However the tasks are not exactly the same. One difference is that the answer is not a vocabulary item or text span, but a single input item. This suggests the use of Pointer Networks (Vinyals, Fortunato, and Jaitly 2015). We obtain the Pointer Attentive Reader for implicit argument prediction. Another difference is that more than one argument may be missing in a predicate-argument tuple. In this case we want the model to reason over the whole document to derive a more informative query. We do this through a multi-hop extension, taking inspiration from multi-hop memory networks (Sukhbaatar et al. 2015). Our model shows good performance on an argument cloze task as well as on a nominal implicit argument prediction task.

2 Related Work

Recent work on implicit arguments started from Gerber and Chai (2010) and Ruppenhofer et al. (2010). Gerber and Chai (2010) constructed a dataset (**G&C**) by selecting 10 nominal predicates and labeling implicit arguments of these predicates in the NomBank (Meyers et al. 2004) corpus manually. The resulting dataset is quite small, consisting of approximately 1000 examples. They also proposed a linear classifier for the task. Gerber and Chai (2012) added more features and performed cross validation on the dataset, leading to better results. Ruppenhofer et al. (2010) also introduced an implicit argument dataset by annotating several chapters of fiction (**SemEval-2010**), which is even smaller (about 500 examples) and more complex than Gerber and Chai (2010). There has since been much follow-up work proposing new methods for G&C (Laparra and Rigau 2013a; Schenk and Chiarcos 2016; Do, Bethard, and Moens 2017) and SemEval-2010 (Silberer and Frank 2012; Laparra and Rigau 2013b; Chiarcos and Schenk 2015). To overcome the size limitation, several methods for creating additional training data have been proposed. Feizabadi and Padó (2015) combined the two datasets, using one as out-of-domain training data for another. Roth and Frank (2015) identified new instances of implicit arguments by aligning

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

monolingual comparable texts, however the size is still similar to that of G&C and SemEval-2010. Schenk and Chiarcos (2016) proposed using text with automatically labeled semantic roles to learn protypical fillers. Both Silberer and Frank (2012) and Cheng and Erk (2018) used coreference information to obtain additional training data. Silberer and Frank (2012) used datasets with manually annotated coreference as additional training data. Cheng and Erk (2018) generated large amounts of training data by using automatically produced coreference labels. They also introduced an additional dataset for testing, which has manually annotated coreference (Hovy et al. 2006) but is automatically manipulated to simulate implicit arguments. We adopt the data generation schema from Cheng and Erk (2018) as the scale allows training of complex neural models. We evaluate our model on the G&C dataset, and compare to models from Gerber and Chai (2012) and Cheng and Erk (2018), which have obtained the best performance on the G&C dataset (discussed in Section 5.3). Recently, O'Gorman et al. (2018) introduced a new AMR corpus with annotation for more than 2000 implicit arguments. While the data is not available yet, it will significantly extend the amount of naturally occurring test data once it is available.

In this paper we draw on recent progress in reading comprehension and memory networks, for the task of implicit argument prediction. Hermann et al. (2015) first introduced neural models to reading comprehension tasks by creating a large cloze-like dataset from news articles paired with human-written summaries. They proposed an Attentive Reader model that used an attention mechanism to reason over the document and query pair. Since then there has been much follow-up work on new datasets (Hill et al. 2016; Rajpurkar et al. 2016; Welbl, Stenetorp, and Riedel 2017) and new models (Chen, Bolton, and Manning 2016; Seo et al. 2017; Dhingra et al. 2017). Another related line of work 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.

We also draw on pointer networks in that we view implicit argument prediction as a pointer to a previous mention of an entity. Vinyals, Fortunato, and Jaitly (2015) first proposed Pointer Networks as a variant of the conventional sequence-to-sequence model 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 of tasks, including Question Answering (Xiong, Zhong, and Socher 2017) and Machine Comprehension (Wang and Jiang 2017).

3 Task Setup

The implicit argument prediction task, as first introduced by Gerber and Chai (2010), is to identify the correct filler for an implicit argument role of a predicate, given the explicit arguments of the same predicate and a list of candidates. This task requires a lot of human effort in annotation, and the existing human-annotated datasets are too small for the use of neural models. The argument cloze task proposed by Cheng and Erk (2018) overcame this difficulty by automatically generating large-scale data for training. The cloze task, as shown in Figure 1, aims to simulate natural occurrences of implicit arguments, and can be briefly described as follows.

Manville Corp. said it will build a $ 24 million power plant to provide electricity to its Igaras pulp and paper mill in Brazil .

The company said the plant will ensure that it has adequate energy for the mill and will reduce the mill's energy costs .

(a) A piece of raw text from OntoNotes corpus.

x_0 = The company x_1 = mill x_2 = power plant

e_0: (*build-pred*, x_0-*subj*, x_2-*dobj*, –)
e_1: (*provide-pred*, –, *electricity-dobj*, *??-prep_to*)
e_2: (*ensure-pred*, x_2-*subj*, –, –)
e_3: (*has-pred*, x_0-*subj*, *energy-dobj*, x_1-*prep_for*)
e_4: (*reduce-pred*, x_2-*subj*, *cost-dobj*, –)

(b) An example of argument cloze task.

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 when viewed as document-query pair.

Figure 1: Example of the argument cloze task and how to view it as reading comprehension. (Part 1a and 1b modified from Cheng and Erk (2018).)

Given a piece of text with dependency parses and coreference chains (x_0~x_2), a sequence of events (predicate-argument tuples, e_0~e_4) are extracted from dependency relations[2]. Then, one argument (i.e., *prep_to* of e_1) that belongs to a coreference chain (x_1) with at least two mentions is randomly selected and removed. The model is asked to pick the removed argument from all coreference chains appearing in the text (Figure 1b).

However, in both manually annotated implicit argument datasets (Gerber and Chai 2010; Ruppenhofer et al. 2010), only preceding mentions are considered as ground truth fillers, so the example in Figure 1b is very dissimilar to naturally occurring implicit arguments. Therefore, to make the argument cloze task closer to the natural task, we 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 1c. For an event (e_4) with a missing argument (*subj*), we concatenate the predicates and arguments of all preceding events (e_0~e_3) and view this 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

[2]The event structure, in which arguments are encoded positionally after the predicate, follows common practice in recent literature of narrative schema (Pichotta and Mooney 2016)

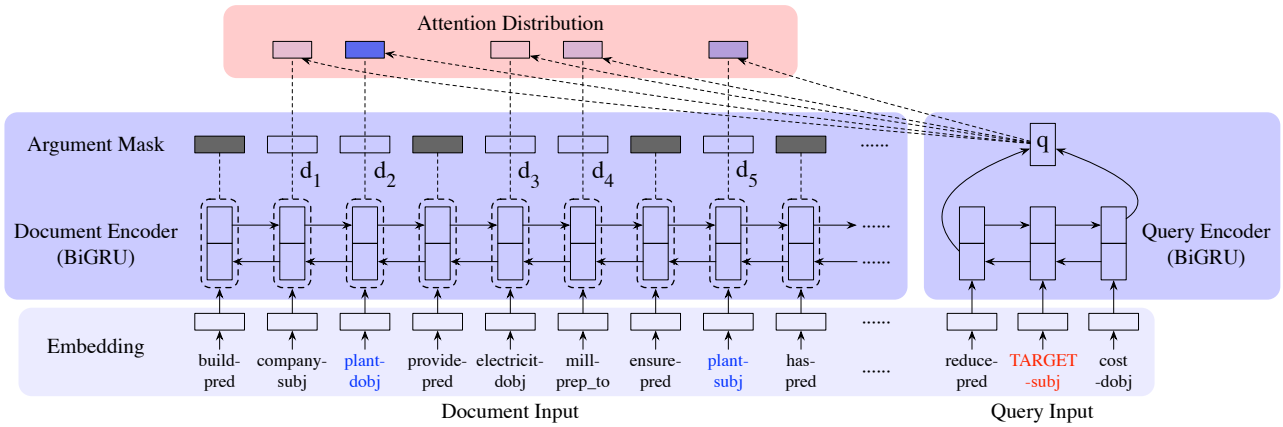

Figure 2: Pointer Attentive Reader. The **document encoder** produces a context-aware embedding for each argument mention via a BiGRU. The **query encoder**, similar to the document encoder, concatenate the last forward and backward hidden state to produce a single query vector. An **attention distribution** is computed from the query vector and all argument mention embeddings, which is then used as a pointer to select one filler for the missing argument in the query.

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 the example.

4 Model

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, Bolton, and Manning 2016; Seo et al. 2017; Dhingra et al. 2017) can be viewed as extending the Attentive Reader model by Hermann et al. (2015). The Attentive Reader 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, Fortunato, and Jaitly 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 2.

4.1 Pointer Attentive Reader

Embedding The document input and the query input, as discussed in Section 3, are both sequences of event components, represented as $[x^d_1, \ldots, x^d_{|D|}]$ and $[x^q_1, \ldots, x^q_{|Q|}]$ 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]:

$$\begin{aligned} s_t &= \mathbf{v}^T \cdot \tanh(\mathbf{W}[\mathbf{d_t}, \mathbf{q}]) \\ a_t &= \text{softmax}(s_t) \end{aligned} \tag{1}$$

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.

Training 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 our implicit argument prediction task (as in the example in Figure

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

1c). Therefore, we train the model to maximize the "maximum correct" attention score. That is, with a list of attention scores $\mathbf{a} = [a_1, a_2, \ldots, a_T] \in \mathcal{R}^T$, and a binary answer mask $\mathbf{m_c} \in \mathcal{R}^T$ which has 1s for correct answer positions (e.g., *plant-dobj* and *plant-subj* in Figure 1c) and 0s elsewhere, we train the model with the following negative log likelihood (NLL) loss function:

$$L = -\log(\max(\mathbf{a} \circ \mathbf{m_c})) \quad (2)$$

where $\circ$ is element-wise multiplication.

4.2 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 3, 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: *build-pred company-subj plant-dobj provide-pred electricity-dobj mill-prep_to ensure-pred plant-subj*
Query 1: *has-pred* ***MISS-subj*** *energy-dobj TARGET-prep_for*

Document 2: *build-pred company-subj plant-dobj provide-pred electricity-dobj mill-prep_to ensure-pred* plant-*subj*
Query 2: *has-pred TARGET-subj energy-dobj* ***MISS-prep_for***

Figure 3: Document-Query example 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 3, 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 shows the 2-hop Pointer Attentive Reader model.

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 $[\mathbf{d_1}, \mathbf{d_2}, \ldots, \mathbf{d_T}]$:

$$\begin{aligned} s'_t &= \mathbf{v'}^T \cdot \tanh(\mathbf{W'}[\mathbf{d_t}, \mathbf{q}]) \\ a'_t &= \text{softmax}(s'_t) \\ \mathbf{o_1} &= \Sigma_{t=1}^{T} a'_t \cdot \mathbf{d_t} \\ \mathbf{q_1} &= \mathbf{o_1} + \mathbf{q} \end{aligned} \quad (3)$$

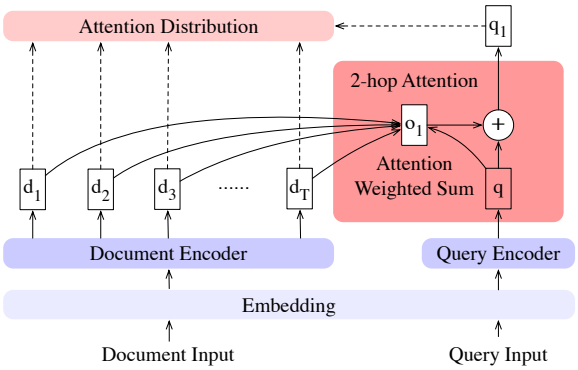

Figure 4: 2-hop Pointer Attentive Reader. The query vector $\mathbf{q}$ is first updated by an attention weighted sum $\mathbf{o_1}$ from all argument embeddings in the document, before used to compute the final attention distribution.

where $\mathbf{W'}$ and $\mathbf{v'}$ are learned parameters. Then in Equation 1 we use $\mathbf{q_1}$ instead of $\mathbf{q}$ to compute the final attention scores.

In this paper we only experiment with 2-hop attention. However the model can be easily extended to k-hop ($k > 2$) attention models.

Extra 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. (In the examples in Figure 3, the distribution would have 0.5 for both *company-subj* and *mill-prep_to*.) Then we can add the KL-divergence between the actual attention scores and the desired distribution to the loss function in Equation 2.

5 Empirical Results

5.1 Training Data

Preprocessing We construct a large scale training dataset from the full English Wikipedia corpus. We retrieve each document from the 20160901 dump of English Wikipedia[4], and split it into paragraphs using the WikiExtractor tool[5].

We run Stanford CoreNLP (Manning et al. 2014) to obtain dependency parses and coreference chains of each paragraph,[6] from which we extract a sequence of events and entities as demonstrated in Figure 1b, after lemmatizing all verbs and arguments, incorporating negation and particles to verbs, and normalizing passive constructions. We downsample the most frequent verbs (with counts over 100,000)

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

[5]https://github.com/attardi/wikiextractor.

[6]The coreference chains are used to make training data, but are not handed to the model.

by a ratio proportional to the square root of their counts, then construct 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 1c). This leads to approximately 25 million document-query pairs in the training dataset. This dataset is used to train models for both evaluation tasks discussed below.

Initialization and Hyperparameters For training the Pointer Attentive Reader model, we initialize the embedding layer with event-based word2vec embeddings, following Cheng and Erk (2018). (The embedding vectors for placeholder tokens 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, Hazan, and Singer 2011) to minimize the negative log-likelihood loss as defined in Equation 2 with a learning rate of 0.01. The 2-hop Pointer Attentive Reader model is trained with the same set of hyperparameters.

5.2 Evaluation on OntoNotes Dataset

Our main evaluation is on the argument cloze task using the OntoNotes datasets of Cheng and Erk (2018). The datasets are large and provide clean test data, as they are based on gold syntax and coreference annotation. The two datasets are ON-SHORT and ON-LONG, where the latter consists of considerably longer documents. We modify their data generation pipeline[7] as discussed in Section 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 our 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 1c), 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 1.

	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 1: Statistics of the OntoNotes datasets.

We compare our 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 Cheng and Erk (2018).

[7] https://github.com/pxch/event_imp_arg

Results The evaluation results are shown in Table 2. We can see that the Pointer Attentive Reader outperforms the previously best EVENTCOMP model by a large margin, especially on the harder ON-LONG dataset. Cheng and Erk (2018) found that entity salience features, that is, numbers of different types of mentions in a coreference chain, greatly improves the performance of their EVENTCOMP model. We have also tried to add such features to our 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 2: Evaluation on the 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 5. 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 our model is able to capture both semantic content of events and salience information of entities.

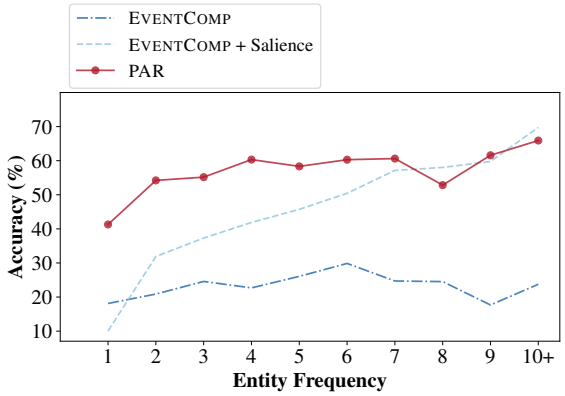

Figure 5: Performance of EVENTCOMP, with and without entity salience, and PAR, by entity frequency (length of coreference chain) of the removed argument, on ON-LONG.

Evaluation on Multiple Implicit Arguments To test our model's ability to predict multiple implicit arguments of the same predicate (Section 4.2), we extract subsets from

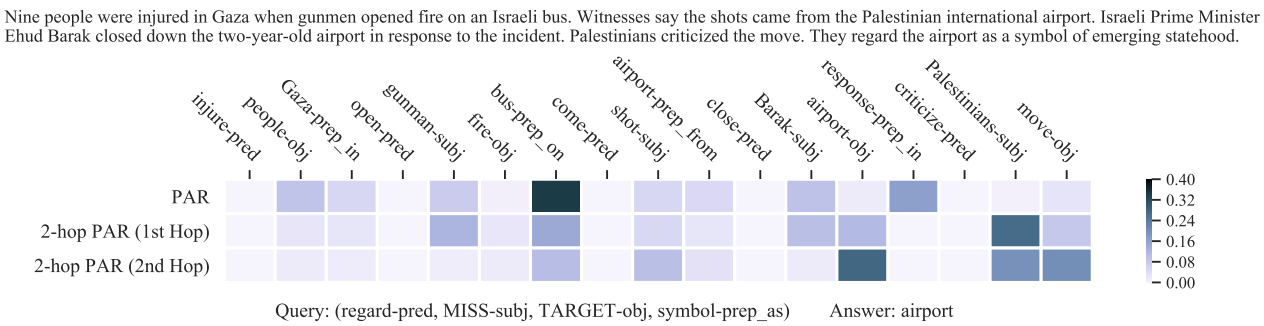

Figure 6: An example from the OntoNotes dataset (`english/bn/cnn_0019`) with multiple implicit arguments, and the attention scores computed by PAR and 2-hop PAR. While PAR fails on this example, 2-hop model succeeds from a more informative query vector when the first hop attends to other missing arguments of the query.

both the ON-SHORT and ON-LONG datasets by selecting queries with 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 3. 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 3: Evaluation on subsets of the OntoNotes datasets with more than one missing argument in the query.

Table 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 document. We also trained a 2-hop model with extra supervision on the first hop of attention scores, as discussed in Section 4.2, but it does not provide much benefit in this experiment. Figure 6 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.

5.3 Evaluation on G&C Dataset

The implicit argument dataset by Gerber and Chai (2010 2012) is a very small dataset with 966 annotated implicit arguments, comprising only 10 nominal predicates. Still it is currently the largest available dataset of naturally occurring implicit arguments.

The task is, for each missing argument, to either choose a filler from a list of candidates or to leave the argument unfilled. The candidates for each missing argument position consists of all core arguments labeled by PropBank (Palmer, Gildea, and Kingsbury 2005) or NomBank (Meyers et al. 2004) within a two-sentence candidate window (i.e., the current sentence and the preceding two sentences). An example is shown below:

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

where *Citicorp* is the implicit subject of *sale*, and *91-day commercial paper* is the implicit object of *sale*.

There are two obstacles to applying our Pointer Attentive Reader to the task. First, the number of missing argument positions (3737) is much larger than the number of gold implicit arguments, making the dataset highly biased. Whether a particular argument position is typically filled is mostly predicate-specific, and the size of dataset makes it hard to train a complex neural model. This problem was also noted by Cheng and Erk (2018), who trained a simple fill / no-fill classifier with a small subset of shallow lexical features used originally by Gerber and Chai (2012). We adapt the same idea to overcome the problem. This also makes our results comparable to Cheng and Erk (2018).

Second, our 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 our 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 our Pointer Attentive Reader 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 sentitive to other hyperparameters, and we cannot find a single set of hyperparameters that works best for all models. Therefore, we report our results as an average of 5 runs with slightly different hyperparameter settings. [8]

Results The evaluation results are presented in Table 4. The GCAUTO and EVENTCOMP results are from Cheng and Erk (2018); GCAUTO is a reimplementation of Gerber and Chai (2012) without gold features. EVENTCOMP* evaluates the EVENTCOMP model in a condition that masks out some candidates to make it a fair comparison with our PAR model, as discussed above. Note that GCAUTO, EVENTCOMP and EVENTCOMP* all have an intrinsic advantage over the PAR model as they exploit event information from the whole document to make the prediction, while our new model 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: Evaluation on the G&C dataset.

The performance of the plain PAR model is already comparable to the GCAUTO baseline. 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 Section 5.2 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 F1 score of 48.3, on par with EVENTCOMP*, the comparably evaluated EVENTCOMP. Figure 7 shows the attention scores of PAR and 2-hop PAR on the previous example, to demonstrate the power of 2-hop inference on multi-argument cases.

[8]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.

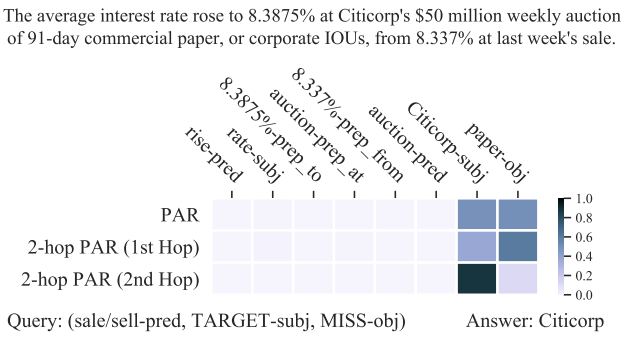

Figure 7: A G&C example with multiple implicit arguments, and the attention scores computed by PAR and 2-hop PAR. While the 2-hop model attends more to the non-target missing argument (paper-obj) on the first hop, it successfully points to the target argument in the second hop.

6 Conclusion

In this paper we have framed 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 have introduced the Pointer Attentive Reader model for implicit argument prediction. On an argument cloze task, the Pointer Attentive Reader beats the previous best model by a large margin, showing good performance on short and long texts, and on 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 a small dataset of naturally occurring nominal predicates. Our code is available at https://github.com/pxch/imp_arg_rc.

In this paper, we have formulated the implicit argument prediction as a task of selecting a mention of an argument, ignoring coreference. In future work, we plan to adapt other widely used reading comprehension models, like BiDAF (Seo et al. 2017), to our task. Another interesting direction is to model coreference latently, through self-attention during the computation of embeddings for the document. We are also interested in integrating implicit argument reasoning in actual reading comprehension. Because argument cloze can be viewed as a variant of reading comprehension, models trained on argument cloze can be straightforwardly integrated into models for reading comprehension.

Acknowledgments

This research was supported by NSF grant IIS 1523637 and by the DARPA AIDA program under AFRL grant FA8750-18-2-0017. We acknowledge the Texas Advanced Computing Center for providing grid resources that contributed to these results, and some results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation. We would like to thank the anonymous reviewers for their valuable feedback.

References

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

Cheng, P., and Erk, K. 2018. 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)*, volume 1, 831–840.

Chiarcos, C., and Schenk, N. 2015. 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*, 178–187.

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

Dhingra, B.; Liu, H.; Yang, Z.; Cohen, W.; and Salakhutdinov, R. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1832–1846.

Do, Q. N. T.; Bethard, S.; and Moens, M.-F. 2017. 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)*, 90–99.

Duchi, J.; Hazan, E.; and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Etzioni, O.; Banko, M.; Soderland, S.; and Weld, D. S. 2007. Open information extraction from the web. In *IJCAI*, 2670–2676.

Feizabadi, P. S., and Padó, S. 2015. Combining seemingly incompatible corpora for implicit semantic role labeling. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, 40–50.

Gerber, M., and Chai, J. 2010. Beyond NomBank: A study of implicit arguments for nominal predicates. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, 1583–1592.

Gerber, M., and Chai, J. Y. 2012. Semantic role labeling of implicit arguments for nominal predicates. *Computational Linguistics* 38(4).

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *NIPS*, 1693–1701.

Hill, F.; Bordes, A.; Chopra, S.; and Weston, J. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR*.

Hovy, E.; Marcus, M.; Palmer, M.; Ramshaw, L.; and Weischedel, R. 2006. OntoNotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*.

Laparra, E., and Rigau, G. 2013a. 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)*, 1180–1189.

Laparra, E., and Rigau, G. 2013b. Sources of evidence for implicit argument resolution. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, 155–166.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412–1421.

Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60.

Meyers, A.; Reeves, R.; Macleod, C.; Szekely, R.; Zielinska, V.; Young, B.; and Grishman, R. 2004. The NomBank project: An interim report. In *Proceedings of the Workshop Frontiers in Corpus Annotation at HLT-NAACL 2004*, 24–31.

O'Gorman, T.; Regan, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; and Palmer, M. 2018. AMR beyond the sentence: the multi-sentence amr corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, 3693–3702.

Palmer, M.; Gildea, D.; and Kingsbury, P. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics* 31(1).

Pichotta, K., and Mooney, R. J. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *AAAI*, 2800–2806.

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

Roth, M., and Frank, A. 2015. Inducing implicit arguments from comparable texts: A framework and its applications. *Computational Linguistics* 41(4):625–664.

Ruppenhofer, J.; Sporleder, C.; Morante, R.; Baker, C.; and Palmer, M. 2010. SemEval-2010 task 10: Linking events and their participants in discourse. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, 45–50.

Schenk, N., and Chiarcos, C. 2016. 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*, 1473–1479.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Silberer, C., and Frank, A. 2012. Casting implicit role linking as an anaphora resolution task. In **SEM 2012*, 1–10.

Stern, A., and Dagan, I. 2014. 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)*, volume 2, 739–744.

Sukhbaatar, S.; Szlam, A.; Weston, J.; and Fergus, R. 2015. End-to-end memory networks. In *NIPS*, 2440–2448.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *NIPS*, 2692–2700.

Wang, S., and Jiang, J. 2017. Machine comprehension using match-lstm and answer pointer. In *ICLR*.

Welbl, J.; Stenetorp, P.; and Riedel, S. 2017. Constructing datasets for multi-hop reading comprehension across documents. *Computing Research Repository* arXiv:1710.06481.

Xiong, C.; Zhong, V.; and Socher, R. 2017. Dynamic coattention networks for question answering. In *ICLR*.