E^3: Entailment-driven Extracting and Editing for Conversational Machine Reading

Victor Zhong
University of Washington
vzhong@cs.washington.edu

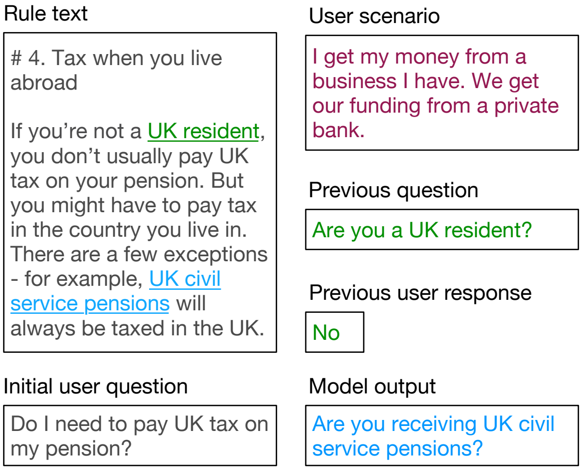
Luke Zettlemoyer
University of Washington
lsz@cs.washington.edu

Abstract

Conversational machine reading systems help users answer high-level questions (e.g. determine if they qualify for particular government benefits) when they do not know the exact rules by which the determination is made (e.g. whether they need certain income levels or veteran status). The key challenge is that these rules are only provided in the form of a procedural text (e.g. guidelines from government website) which the system must read to figure out what to ask the user. We present a new conversational machine reading model that jointly extracts a set of decision rules from the procedural text while reasoning about which are entailed by the conversational history and which still need to be edited to create questions for the user. On the recently introduced ShARC conversational machine reading dataset, our Entailment-driven Extract and Edit network (E^3) achieves a new state-of-the-art, outperforming existing systems as well as a new BERT-based baseline. In addition, by explicitly highlighting which information still needs to be gathered, E^3 provides a more explainable alternative to prior work. We release source code for our models and experiments at https://github.com/vzhong/e3.

1 Introduction

In conversational machine reading (CMR), a system must help users answer high-level questions by participating in an information gathering dialog. For example, in Figure 1 the system asks a series of questions to help the user decide if they need to pay tax on their pension. A key challenge in CMR is that the rules by which the decision is made are only provided in natural language (e.g. the rule text in Figure 1). At every step of the conversation, the system must read the rules text and reason about what has already been said in to formulate the best next question.



Rule text

# 4. Tax when you live abroad

If you're not a UK resident, you don't usually pay UK tax on your pension. But you might have to pay tax in the country you live in. There are a few exceptions - for example, UK civil service pensions will always be taxed in the UK.

User scenario

I get my money from a business I have. We get our funding from a private bank.

Previous question

Are you a UK resident?

Previous user response

No

Initial user question

Do I need to pay UK tax on my pension?

Model output

Are you receiving UK civil service pensions?

Figure 1: A conversational machine reading example. The model is given a rule text document, which contains a recipe of implicit rules (underlined) for answering the initial user question. At the start of the conversation, the user presents a scenario describing their situation. During each turn, the model can ask the user a follow-up question to inquire about missing information, or conclude the dialogue by answering `yes`, `no`, or `irrelevant`. `irrelevant` means that the rule text cannot answer the question. We show previous turns as well as the corresponding inquired rules in green. The scenario is shown in red and in this case does not correspond to a rule. The model inquiry for this turn and its corresponding rule are shown in blue.

We present a new model that jointly reasons about what rules are present in the text and which are already entailed by the conversational history to improve question generation. More specifically, we propose the Entailment-driven Extract and Edit network (E^3). E^3 learns to extract implicit rules in the document, identify which rules are entailed by the conversation history, and edit rules that are not entailed to create follow-up questions to the user. During each turn, E^3 parses the rule text to extract spans in the text that correspond to implicit rules (underlined in Figure 1). Next, the model scores the degree to which each extracted rule is entailed

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2310–2320
Florence, Italy, July 28 - August 2, 2019. ©2019 Association for Computational Linguistics

by the initial user scenario (red in Figure 1) and by previous interactions with the user (green in Figure 1). Finally, the model decides on a response by directly answering the question (yes/no), stating that the rule text does not contain sufficient information to answer the question (irrelevant), or asking a follow-up question about an extracted rule that is not entailed but needed to determine the answer (blue in Figure 1). In the case of inquiry, the model edits an extracted rule into a follow-up question. To our knowledge, E^3 is the first extract-and-edit method for conversational dialogue, as well as the first method that jointly infers implicit rules in text, estimates entailment, inquires about missing information, and answers the question.

We compare E^3 to the previous-best systems as well as a new, strong, BERT-based extractive question answering model (BERTQA) on the recently proposed ShARC CMR dataset (Saeidi et al., 2018). Our results show that E^3 is more accurate in its decisions and generates more relevant inquiries. In particular, E^3 outperforms the previous-best model by 5.7% in micro-averaged decision accuracy and 4.3 in inquiry BLEU4. Similarly, E^3 outperforms the BERTQA baseline by 4.0% micro-averaged decision accuracy and 2.4 in inquiry BLEU4. In addition to outperforming previous methods, E^3 is explainable in the sense that one can visualize what rules the model extracted and how previous interactions and inquiries ground to the extracted rules. We release source code for E^3 and the BERTQA model at https://github.com/vzhong/e3.

2 Related Work

Dialogue tasks. Recently, there has been growing interest in question answering (QA) in a dialogue setting (Choi et al., 2018; Reddy et al., 2019). CMR (Saeidi et al., 2018) differs from dialogue QA in the domain covered (regulatory text vs Wikipedia). A consequence of this is that CMR requires the interpretation of complex decision rules in order to answer high-level questions, whereas dialogue QA typically contains questions whose answers are directly extractable from the text. In addition, CMR requires the formulation of free-form follow-up questions in order to identify whether the user satisfies decision rules, whereas dialogue QA does not. There has also been significant work on task-oriented dialogue, where the system must inquire about missing information in order to help the user achieve a goal (Williams et al., 2013; Henderson et al., 2014; Mrkšić et al., 2017; Young et al., 2013). However, these tasks are typically constrained to a fixed ontology (e.g. restaurant reservation), instead of a latent ontology specified via natural language documents.

Dialogue systems. One traditional approach for designing dialogue systems divides the task into language understanding/state-tracking (Mrkšić et al., 2017; Zhong et al., 2018), reasoning/policy learning (Su et al., 2016), and response generation (Wen et al., 2015). The models for each of these subtasks are then combined to form a full dialogue system (Young et al., 2013; Wen et al., 2017). The previous best system for ShARC (Saeidi et al., 2018) similarly breaks the CMR task into subtasks and combines hand-designed sub-models for decision classification, entailment, and follow-up generation. In contrast, the core reasoning (e.g. non-editor) components of E^3 are jointly trained, and does not require complex hand-designed features.

Extracting latent rules from text. There is a long history of work on extracting knowledge automatically from text (Moulin and Rousseau, 1992). Relation extraction typically assumes that there is a fixed ontology onto which extracted knowledge falls (Mintz et al., 2009; Riedel et al., 2013). Other works forgo the ontology by using, for example, natural language (Angeli and Manning, 2014; Angeli et al., 2015). These extractions from text are subsequently used for inference over a knowledge base (Bordes et al., 2013; Dettmers et al., 2018; Lin et al., 2018) and rationalizing model predictions (Lei et al., 2016). Our work is more similar with the latter type in which knowledge extracted are not confined to a fixed ontology and instead differ on a document basis. In addition, the rules extracted by our model are used for inference over natural language documents. Finally, these rules provide rationalization for the model's decision making, in the sense that the user can visualize what rules the model extracted and which rules are entailed by previous turns.

3 Entailment-driven Extract and Edit network

In conversational machine reading, a system reads a document that contains a set of implicit decision

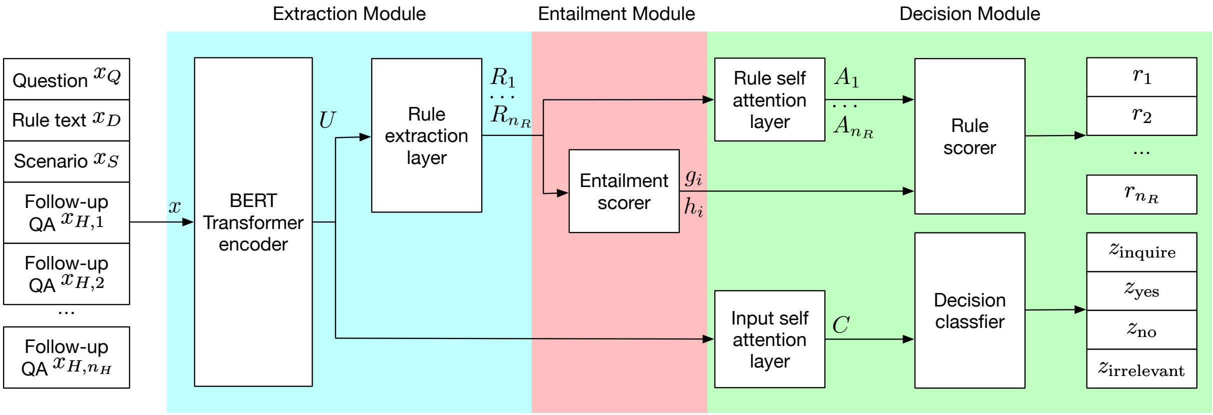


Figure 2: The Entailment-driven Extract and Edit network.

rules. The user presents a scenario describing their situation, and asks the system an underspecified question. In order to answer the user's question, the system must ask the user a series of follow-up questions to determine whether the user satisfies the set of decision rules.

The key challenges in CMR are to identify implicit rules present in the document, understand which rules are necessary to answer the question, and inquire about necessary rules that are not entailed by the conversation history by asking follow-up questions. The three core modules of E^3, the extraction, entailment, and decision modules, combine to address these challenges. Figure 2 illustrates the components of E^3.

For ease of exposition, we describe E^3 for a single turn in the conversation. To make the references concrete in the following sections, we use as an example the inputs and outputs from Figure 1. This example describes a turn in a conversation in which the system helps the user determine whether they need to pay UK taxes on their pension.

3.1 Extraction module

The extraction module extracts spans from the document that correspond to latent rules. Let x_D, x_Q, x_S, $x_{H,i}$ denote words in the rule text, question, scenario, and the inquiry and user response during the ith previous turn of the dialogue after N turns have passed. We concatenate these inputs into a single sequence $x = [x_Q; x_D; x_S; x_{H,1}; \cdots x_{H,N}]$ joined by sentinel tokens that mark the boundaries of each input. To encode the input for the extraction module, we use BERT, a transformer-based model (Vaswani et al., 2017) that achieves consistent gains on a variety of NLP tasks (Devlin et al., 2019). We encode x using the BERT encoder, which first converts words into word piece tokens (Wu et al., 2016), then embeds these tokens along with their positional embeddings and segmentation embeddings. These embeddings are subsequently encoded via a transformer network, which allows for inter-token attention at each layer. Let n_x be the number of tokens in the concatenated input x and d_U be the output dimension of the BERT encoder. For brevity, we denote the output of the BERT encoder as $U = \text{BERT}(x) \in \mathbb{R}^{n_x \times d_U}$ and refer readers to Devlin et al. (2019) for detailed architecture.

In order to extract the implicit decision rules from the document, we compute a start score α_i and an end score β_i for each ith token as

$$\alpha_i = \sigma\left(W_\alpha U_i + b_\alpha\right) \in \mathbb{R} \quad (1)$$

$$\beta_i = \sigma\left(W_\beta U_i + b_\beta\right) \in \mathbb{R} \quad (2)$$

where $W_\alpha, W_\beta \in \mathbb{R}^{d_U}$, $b_\alpha, b_\beta \in \mathbb{R}$, and σ is the sigmoid function.

For each position s_i where α_i is larger than some threshold τ, we find the closest proceeding position $e_i \geq s_i$ where $\beta_{e_i} > \tau$. Each pair (s_i, e_i) then forms an extracted span corresponding to a rule R_i expressed in the rule text. In the example in Figure 1, the correct extracted spans are "UK resident" and "UK civil service pensions".

For the ith rule, we use self-attention to build a representation $\overline{A}_i$ over the span (s_i, e_i).

$$\overline{\gamma}_k = W_\gamma U_k + b_\gamma \in \mathbb{R}, s_i \leq k \leq e_i \quad (3)$$

$$\gamma_k = \text{softmax}\left(\overline{\gamma}\right)_k \in \mathbb{R}, s_i \leq k \leq e_i \quad (4)$$

$$\overline{A}_i = \sum_{k=s_i}^{e_i} \gamma_k U_k \in \mathbb{R}^{d_U} \quad (5)$$

where $W_\gamma \in \mathbb{R}^{d_U}$ and $b_\gamma \in \mathbb{R}$. Here, $\overline{\gamma}_k, \gamma_k$ are respectively the unnormalized and normalized scores for the self-attention layer.

Let n_R denote the number spans in the rule text, each of which corresponds to a ground truth rule. The rule extraction loss is computed as the sum of the binary cross entropy losses for each rule R_i.

$$L_{\text{re}} = \sum_i^{n_R} L_{\text{start},i} + L_{\text{end},i} \tag{6}$$

Let n_D denote the number of tokens in the rule text, s_i, e_i the ground truth start and end positions for the ith rule, and $\mathbb{1}_f$ the indicator function that returns 1 if and only if the condition f holds. Recall from Eq (1) that α_j and β_j denote the probabilities that token j is the start and end of a rule. The start and end binary cross entropy losses for the ith rule are computed as

$$L_{\text{start},i} = -\sum_j^{n_D} \mathbb{1}_{j=s_i} \log(\alpha_j) + \mathbb{1}_{j\neq s_i} \log(1-\alpha_j)$$

$$L_{\text{end},i} = -\sum_j^{n_D} \mathbb{1}_{j=e_i} \log(\beta_j) + \mathbb{1}_{j\neq e_i} \log(1-\beta_j)$$

3.2 Entailment module

Given the extracted rules $R = \{R_1, \cdots R_{n_R}\}$, the entailment module estimates whether each rule is entailed by the conversation history, so that the model can subsequently inquire about rules that are not entailed. For the example in Figure 1, the rule "UK resident" is entailed by the previous inquiry "Are you a UK resident". In contrast, the rule "UK civil service pensions" is not entailed by either the scenario or the conversation history, so the model needs to inquire about it. In this particular case the scenario does not entail any rule.

For each extracted rule, we compute a score that indicates the extent to which this particular rule has already been discussed in the initial scenario S and in previous turns Q. In particular, let $N(R_i, S)$ denote the number of tokens shared by R_i and S, $N(R_i)$ the number of tokens in R_i, and $N(S)$ the number of tokens in S. We compute the scenario entailment score g_i as

$$\text{pr}(R_i, S) = \frac{N(R_i, S)}{N(R_i)} \tag{7}$$

$$\text{re}(R_i, S) = \frac{N(R_i, S)}{N(S)} \tag{8}$$

$$g_i = \text{f1}(R_i, S) = \frac{2\text{pr}(R_i, S)\text{re}(R_i, S)}{\text{pr}(R_i, S) + \text{re}(R_i, S)} \tag{9}$$

where pr, re, and f1 respectively denote the precision, recall, and F1 scores. We compute a similar score to represent the extent to which the rule R_i has been discussed in previous inquiries. Let Q_k denote tokens in the kth previous inquiry. We compute the history entailment score h_i between the extracted rule R_i and all n_H previous inquiries in the conversation history as

$$h_i = \max_{k=1,\cdots n_H} \text{f1}(R_i, Q_k) \tag{10}$$

The final representation of the ith rule, A_i, is then the concatenation of the span self-attention and the entailment scores.

$$A_i = [\overline{A}_i; g_i; h_i] \in \mathbb{R}^{d_U+2} \tag{11}$$

where $[x; y]$ denotes the concatenation of x and y. We also experiment with embedding and encoding similarity based approaches to compute entailment, but find that this F1 approach performs the best. Because the encoder utilizes cross attention between different components of the input, the representations U and $\overline{A}_i$ are able to capture notions of entailment. However, we find that explicitly scoring entailment via the entailment module further discourages the model from making redundant inquiries.

3.3 Decision module

Given the extracted rules R and the entailment-enriched representations for each rule A_i, the decision module decides on a response to the user. These include answering `yes`/`no` to the user's original question, determining that the rule text is `irrelevant` to the question, or inquiring about a rule that is not entailed but required to answer the question. For the example in Figure 1, the rule "UK civil service pensions" is not entailed, hence the correct decision is to ask a follow-up question about whether the user receives this pension.

We start by computing a summary C of the input using self-attention

$$\overline{\phi}_k = W_\phi U_k + b_\phi \in \mathbb{R} \tag{12}$$

$$\phi_k = \text{softmax}\left(\overline{\phi}\right)_k \in \mathbb{R} \tag{13}$$

$$C = \sum_{k=s_i}^{e_i} \phi_k U_k \in \mathbb{R}^{d_U} \tag{14}$$

where $W_\phi \in \mathbb{R}^{d_U}$, $b_\phi \in \mathbb{R}$, and $\overline{\phi}$, ϕ are respectively the unnormalized and normalized self-attention weights. Next, we score the choices `yes`, `no`, `irrelevant`, and `inquire`.

$$z = W_z C + b_z \in \mathbb{R}^4 \tag{15}$$

where z is a vector containing a class score for each of the `yes`, `no`, `irrelevant`, and `inquire` decisions.

For inquiries, we compute an inquiry score r_i for each extracted rule R_i.

$$r_i = W_z A_i + b_z \in \mathbb{R} \tag{16}$$

where $W_z \in \mathbb{R}^{d_U+2}$ and $b_z \in \mathbb{R}$. Let k indicate the correct decision, and i indicate the correct inquiry, if the model is supposed to make an inquiry. The decision loss is

$$\begin{aligned} L_{\text{dec}} &= -\log \text{softmax}(z)_k \\ &\quad -\mathbb{1}_{k=\text{inquire}} \log \text{softmax}(r)_i \end{aligned} \tag{17}$$

During inference, the model first determines the decision $d = \text{argmax}_k z_k$. If the decision d is `inquire`, the model asks a follow-up question about the ith rule such that $i = \text{argmax}_j r_j$. Otherwise, the model concludes the dialogue with d.

Rephrasing rule into question via editor. In the event that the model chooses to make an inquiry about an extracted rule R_i, R_i is given to an subsequent editor to rephrase into a follow-up question. For the example in 1, the editor edits the span "UK civil service pensions" into the follow-up question "Are you receiving UK civil service pensions?" Figure 3 illustrates the editor.

The editor takes as input $x_{\text{edit}} = [R_i; x_D]$, the concatenation of the extracted rule to rephrase R_i and the rule text x_D. As before, we encode using a BERT encoder to obtain $U_{\text{edit}} = \text{BERT}(x_{\text{edit}})$. The encoder is followed by two decoders that respective generate the pre-span edit $R_{i,\text{pre}}$ and post-span edit $R_{i,\text{post}}$. For the example in Figure 1, given the span "UK civil service pensions", the pre-span and post span edits that form the question "Are you receiving UK civil service pensions?" are respectively "Are you receiving" and "?"

To perform each edit, we employ an attentive decoder (Bahdanau et al., 2015) with Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997). Let h_t denote the decoder state at time t. We compute attention a_t over the input.

$$\begin{aligned} \bar{\zeta}_k &= U_{\text{edit}} h_{t-1} \in \mathbb{R} && (18) \\ \zeta_k &= \text{softmax}(\bar{\zeta})_k \in \mathbb{R} && (19) \\ a_t &= \sum_k \zeta_k U_{\text{edit},k} \in \mathbb{R}^{d_U} && (20) \end{aligned}$$

Let $V \in \mathbb{R}^{n_V \times d_V}$ denote the embedding matrix corresponding to n_V tokens in the vocabulary.

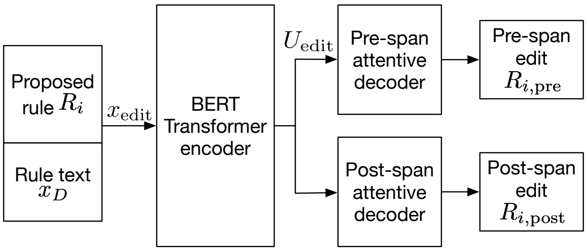

Proposed rule R_i
Rule text x_D
x_{edit}
BERT Transformer encoder
U_{edit}
Pre-span attentive decoder
Post-span attentive decoder
Pre-span edit $R_{i,\text{pre}}$
Post-span edit $R_{i,\text{post}}$

Figure 3: The editor of E^3.

To generate the tth token w_t, we use weight tying between the output layer and the embedding matrix (Press and Wolf, 2017).

$$\begin{aligned} v_t &= \text{embed}(V, w_{t-1}) && (21) \\ h_t &= \text{LSTM}\left([v_t; a_t], h_{t-1}\right) \in \mathbb{R}^{d_U} && (22) \\ o_t &= W_o[h_t; a_t] + b_o \in \mathbb{R}^{d_V} && (23) \\ p(w_t) &= \text{softmax}(V o_t) \in \mathbb{R}^{n_V} && (24) \\ w_t &= \text{argmax}_k p(w_t)_k && (25) \end{aligned}$$

We use a separate attentive decoder to generate the pre-span edit $R_{i,\text{pre}}$ and the post-span edit $R_{i,\text{post}}$. The decoders share the embedding matrix and BERT encoder but do not share other parameters. The output of the editor is the concatenation of tokens $[R_{i,\text{pre}}; R_i; R_{i,\text{post}}]$.

The editing loss consists of the sequential cross entropy losses from generating the pre-span edit and the post-span edit. Let n_{pre} denote the number of tokens and $\hat{w}_{t,\text{pre}}$ the tth tokens in the ground truth pre-span edit. The pre-span loss is

$$L_{\text{pre}} = -\sum_t^{n_{\text{pre}}} \log p(\hat{w}_{t,\text{pre}}) \tag{26}$$

The editing loss is then the sum of the pre-span and post-span losses, the latter of which is obtained in a manner similar to Eq (26).

$$L_{\text{edit}} = L_{\text{pre}} + L_{\text{post}} \tag{27}$$

4 Experiment

We train and evaluate the Entailment-driven Extract and Edit network on the ShARC CMR dataset. In particular, we compare our method to three other models. Two of these models are proposed by Saeidi et al. (2018). They are an attentive sequence-to-sequence model that attends to the concatenated input and generates the response token-by-token (Seq2Seq), and a strong hand-engineered pipeline model with submodels for entailment, classification, and generation (Pipeline). For the latter, Saeidi et al. (2018)

Model	Micro Acc.	Macro Acc.	BLEU1	BLEU4	Comb.
Seq2Seq	44.8	42.8	34.0	7.8	3.3
Pipeline	61.9	68.9	**54.4**	34.4	23.7
BERTQA	63.6	70.8	46.2	36.3	25.7
E^3 (ours)	**67.6**	**73.3**	54.1	**38.7**	**28.4**

Table 1: Model performance on the blind, held-out test set of ShARC. The evaluation metrics are micro and macro-averaged accuracy in classifying bewteen the decisions `yes`, `no`, `irrelevant`, and `inquire`. In the event of an inquiry, the generated follow-up question is further evaluated using the BLEU score. In addition to official evaluation metrics, we also show a combined metric ("Comb."), which is the product between the macro-averaged accuracy and the BLEU4 score.

show that these sub-models outperform neural models such as the entailment model by Parikh et al. (2016), and that the combined pipeline outperforms the attentive sequence-to-sequence model. In addition, we propose an extractive QA baseline based on BERT (BERTQA). Similar models achieved state-of-the-art on a variety of QA tasks (Rajpurkar et al., 2016; Reddy et al., 2019). We refer readers to Section A.1 of the appendices for implementation details BERTQA.

4.1 Experimental setup

We tokenize using revtok[1] and part-of-speech tag (for the editor) using Stanford CoreNLP (Manning et al., 2014). We fine-tune the smaller, uncased pretrained BERT model by Devlin et al. (2019) (e.g. `bert-base-uncased`).[2] We optimize using ADAM (Kingma and Ba, 2015) with an initial learning rate of 5e-5 and a warm-up rate of 0.1. We regularize using Dropout (Srivastava et al., 2014) after the BERT encoder with a rate of 0.4.

To supervise rule extraction, we reconstruct full dialogue trees from the ShARC training set and extract all follow-up questions as well as bullet points from each rule text and its corresponding dialogue tree. We then match these extracted clauses to spans in the rule text, and consider these noisy matched spans as supervision for rule extraction. During inference, we use heuristic bullet point extraction[3] in conjunction with spans extracted by the rule extraction module. This results in minor performance improvements ($\sim$ 1% micro/macro acc.) over only relying on the rule extraction module. In cases where one rule fully covers another, we discard the covered shorter rule. Section A.2 details how clause matching is used to obtain noisy supervision for rule extraction.

We train the editor separately, as jointly training with a shared encoder worsens performance. The editor is trained by optimizing L_{edit} while the rest of the model is trained by optimizing $L_{\text{dec}} + \lambda L_{\text{re}}$. We use a rule extraction threshold of $\tau = 0.5$ and a rule extraction loss weight of $\lambda = 400$. We perform early stopping using the product of the macro-averaged accuracy and the BLEU4 score.

For the editor, we use fixed, pretrained embeddings from GloVe (Pennington et al., 2014), and use dropout after input attention with a rate of 0.4. Before editing retrieved rules, we remove prefix and suffix adpositions, auxiliary verbs, conjunctions, determiners, or punctuation. We find that doing so allows the editor to convert some extracted rules (e.g. or sustain damage) into sensible questions (e.g. did you sustain damage?).

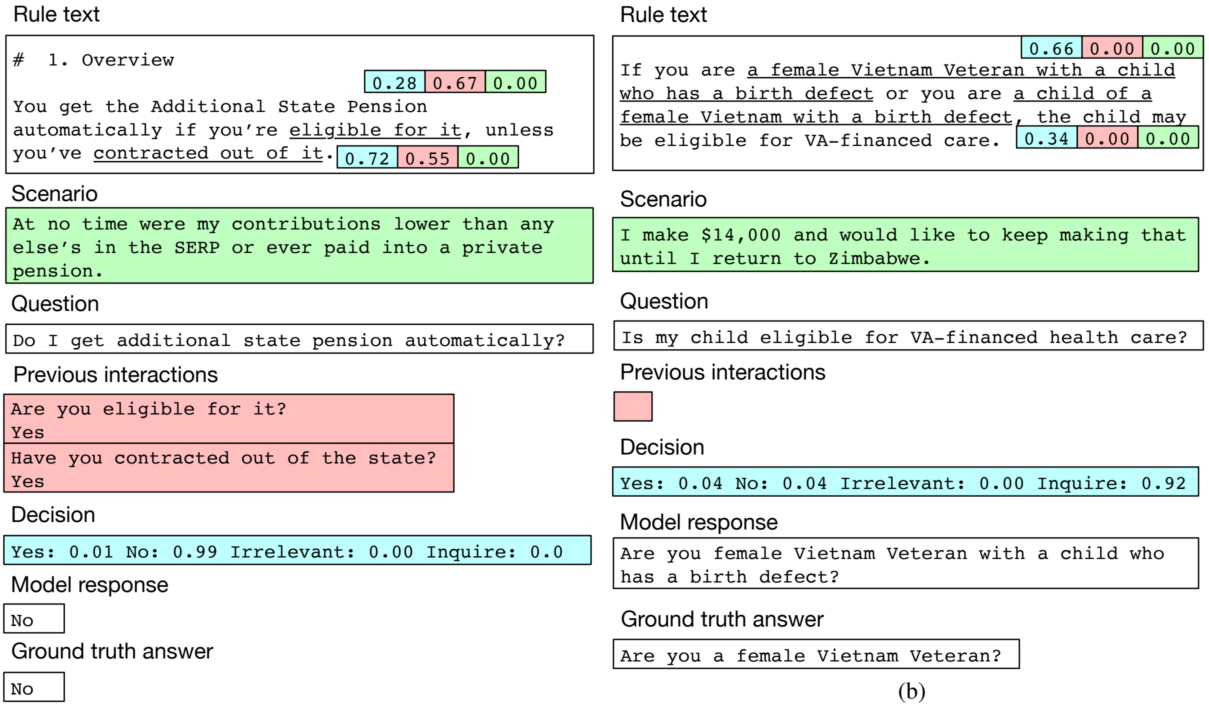
4.2 Results

Our performance on the development and the blind, held-out test set of ShARC is shown in Table 1. Compared to previous results, E^3 achieves a new state-of-the-art, obtaining best performance on micro and macro-averaged decision classification accuracy and BLEU4 scores while maintaining similar BLEU1 scores. These results show that E^3 both answers the user's original question more accurately, and generates more coherent and relevant follow-up questions. In addition, Figure 4 shows that because E^3 explicitly extracts implicit rules from the document, the model's predictions are explainable in the sense that the user can verify the correctness of the extracted rules and observe how the scenario and previous interactions ground to the extracted rules.

[1] https://github.com/jekbradbury/revtok

[2] We use the BERT implementation from https://github.com/huggingface/pytorch-pretrained-BERT

[3] We extract spans from the text that starts with the "*" character and ends with another "*" character or a new line.



Rule text

```
#   1. Overview
                                        0.28 0.67 0.00
You get the Additional State Pension
automatically if you're eligible for it, unless
you've contracted out of it. 0.72 0.55 0.00
```

Scenario

```
At no time were my contributions lower than any
else's in the SERP or ever paid into a private
pension.
```

Question

```
Do I get additional state pension automatically?
```

Previous interactions

```
Are you eligible for it?
Yes
Have you contracted out of the state?
Yes
```

Decision

```
Yes: 0.01 No: 0.99 Irrelevant: 0.00 Inquire: 0.0
```

Model response

```
No
```

Ground truth answer

```
No
```

(a)

Rule text

```
                                    0.66 0.00 0.00
If you are a female Vietnam Veteran with a child
who has a birth defect or you are a child of a
female Vietnam with a birth defect, the child may
be eligible for VA-financed care. 0.34 0.00 0.00
```

Scenario

```
I make $14,000 and would like to keep making that
until I return to Zimbabwe.
```

Question

```
Is my child eligible for VA-financed health care?
```

Previous interactions

Decision

```
Yes: 0.04 No: 0.04 Irrelevant: 0.00 Inquire: 0.92
```

Model response

```
Are you female Vietnam Veteran with a child who
has a birth defect?
```

Ground truth answer

```
Are you a female Vietnam Veteran?
```

(b)

Figure 4: Predictions by E^3. Extracted spans are underlined in the text. The three scores are the inquiry score r_i (blue), history entailment score h_i (red), and scenario entailment score g_i (green) of the nearest extracted span.

Model	Micro Acc.	Macro Acc.	BLEU1	BLEU4	Comb.
E^3	68.0	73.4	66.9	53.7	39.4
-edit	68.0	73.4	53.1	46.2	31.4
-edit, entail	68.0	73.1	50.2	40.3	29.5
-edit, entail, extract (BERTQA)	63.4	70.6	47.4	37.4	23.7

Table 2: Ablation study of E^3 on the development set of ShARC. The ablated variants of E^3 include versions: without the editor; without the editor and entailment module; without the editor, entailment module, and extraction module, which reduces to the BERT for question answering model by Devlin et al. (2019).

4.3 Ablation study

Table 2 shows an ablation study of E^3 on the development set of ShARC.

Retrieval outperforms word generation. BERTQA ("-edit, entail, extract"), which E^3 reduces to after removing the editor, entailment, and extraction modules, presents a strong baseline that exceeds previous results on all metrics except for BLEU1. This variant inquires about spans extracted from the text, which, while more relevant as indicated by the higher BLEU4 score, does not have the natural qualities of a question, hence it has a lower BLEU1. Nonetheless, the large gains of BERTQA over the attentive Seq2Seq model shows that retrieval is a more promising technique for asking follow-up questions than word-by-word generation. Similar findings were reported for question answering by Yatskar (2019).

Extraction of document structure facilitates generalization. Adding explicit extraction of rules in the document ("-edit, entail") forces the model to interpret all rules in the document versus only focusing on extracting the next inquiry. This results in better performance in both decision classification and inquiry relevance compared to the variant that is not forced to interpret all rules.

Modeling entailment improves rule retrieval. The "-edit" model explicitly models whether an extracted rule is entailed by the user scenario and previous turns. Modeling entailment allows the model to better predict whether a rule is entailed,

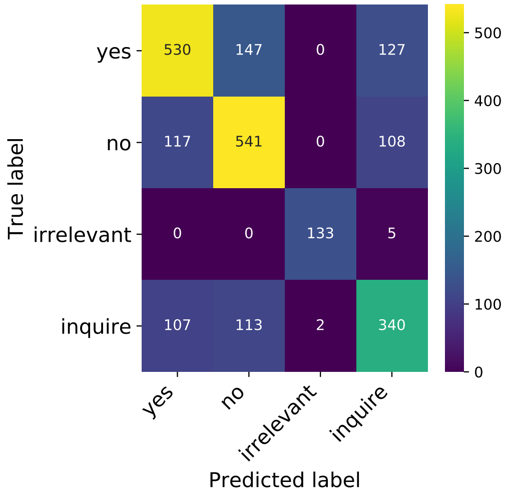


Figure 5: Confusion matrix of decision predictions on the development set of ShARC.

and thus more often inquire about rules that are not entailed. Figure 4a illustrates one such example in which both extracted rules have high entailment score, and the model chooses to conclude the dialogue by answering `no` instead of making further inquiries. Adding entailment especially improves in BLEU4 score, as the inquiries made by the model are more relevant and appropriate.

Editing retrieved rules results in more fluid questions. While E^3 without the editor is able to retrieve rules that are relevant, these spans are not fluent questions that can be presented to the user. The editor is able to edit the extracted rules into more fluid and coherent questions, which results further gains particularly in BLEU1.

4.4 Error analysis

In addition to ablation studies, we analyze errors E^3 makes on the development set of ShARC.

Decision errors. Figure 5 shows the confusion matrix of decisions. We specifically examine examples in which E^3 produces an incorrect decision. On the ShARC development set there are 726 such cases, which correspond to a 32.0% error rate. We manually analyze 100 such examples to identify commons types of errors. Within these, in 23% of examples, the model attempts to answer the user's initial question without resolving a necessary rule despite successfully extracting the rule. In 19% of examples, the model identifies and inquires about all necessary rules but comes to the wrong conclusion. In 18% of examples, the model makes a redundant inquiry about a rule that is entailed. In 17% of examples, the rule text contains ambiguous rules. Figure 4b contains one such example in which the annotator identified the rule "a female Vietnam Veteran", while the model extracted an alternative longer rule "a female Vietnam Veteran with a child who has a birth defect". Finally, in 13% of examples, the model fails to extract some rule from the document. Other less common forms of errors include failures by the entailment module to perform numerical comparison, complex rule procedures that are difficult to deduce, and implications that require world knowledge. These results suggests that improving the decision process after rule extraction is an important area for future work.

Inquiry quality. On 340 examples (15%) in the ShARC development set, E^3 generates an inquiry when it is supposed to. We manually analyze 100 such examples to gauge the quality of generated inquiries. On 63% of examples, the model generates an inquiry that matches the ground-truth. On 14% of examples, the model makes inquires in a different order than the annotator. On 12% of examples, the inquiry refers to an incorrect subject (e.g. "are you born early" vs. "is your baby born early". This usually results from editing an entity-less bullet point ("* born early"). On 6% of examples, the inquiry is lexically similar to the ground truth but has incorrect semantics (e.g. "do you need savings" vs. "is this information about your savings"). Again, this tends to result from editing short bullet points (e.g. "* savings"). These results indicate that when the model correctly chooses to inquire, it largely inquires about the correct rule. They also highlight a difficulty in evaluating CMR — there can be several correct orderings of inquiries for a document.

5 Conclusion

We proposed the Entailment-driven Extract and Edit network (E^3), a conversational machine reading model that extracts implicit decision rules from text, computes whether each rule is entailed by the conversation history, inquires about rules that are not entailed, and answers the user's question. E^3 achieved a new state-of-the-art result on the ShARC CMR dataset, outperforming existing systems as well as a new extractive QA baseline based on BERT. In addition to achieving strong performance, we showed that E^3 provides a more explainable alternative to prior work which do not model document structure.

Acknowledgments

This research was supported in part by the ARO (W911NF-16-1-0121) and the NSF (IIS-1252835, IIS-1562364). We thank Terra Blevins, Sewon Min, and our anonymous reviewers for helpful feedback.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL*.

Gabor Angeli and Christopher D. Manning. 2014. Naturalli: Natural logic inference for common sense reasoning. In *EMNLP*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *EMNLP*.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *AAAI*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *SIGDIAL*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. In *EMNLP*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *EMNLP*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

B. Moulin and D. Rousseau. 1992. Automated knowledge acquisition from regulatory texts. *IEEE Expert*.

Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *ACL*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *ACL*.

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

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. CoQA: A conversational question answering challenge. *TACL*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL*.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. In *EMNLP*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

Pei-Hao Su, Milica Gasic, Nikola Mrkšić, Lina M. Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

Jason D Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *SIGDIAL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Mark Yatskar. 2019. A qualitative comparison of coqa, squad 2.0 and quac. In *NAACL*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive dialogue state tracker. In *ACL*.

A Appendices

A.1 BertQA Baseline

Our BertQA baseline follows that proposed by Devlin et al. (2019) for the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016). Due to the differences in context between ShARC and SQuAD, we augment the input to the BERTQA model in a manner similar to Section 3.1. The distinction here is that we additionally add the decision types "yes", "no", and "irrelevant" as parts of the input such that the problem is fully solvable via span extraction. Similar to Section 3.1, let U denote the BERT encoding of the length-n input sequence. The BERTQA model predicts a start score s and an end score e.

$$s = \text{softmax}(UW_s + b_s) \in \mathbb{R}^n \tag{28}$$

$$e = \text{softmax}(UW_e + b_e) \in \mathbb{R}^n \tag{29}$$

We take the answer as the span (i, j) that gives the highest score $s_i e_j$ such that $j >= i$. Because we augment the input with decision labels, the model can be fully supervised via extraction endpoints.

A.2 Creating noisy supervision for span extraction via span matching

The ShARC dataset is constructed from full dialogue trees in which annotators exhaustively annotate yes/no branches of follow-up questions. Consequently, each rule required to answer the initial user question forms a follow-up question in the full dialogue tree. In order to identify rule spans in the document, we first reconstruct the dialogue trees for all training examples in ShARC. For each document, we trim each follow-up question in its corresponding dialogue tree by removing punctuation and stop words. For each trimmed question, we find the shortest best-match span in the document that has the least edit distance from the trimmed question, which we take as the corresponding rule span. In addition, we extract similarly trimmed bullet points from the document as rule spans. Finally, we deduplicate the rule spans by removing those that are fully covered by a longer rule span. Our resulting set of rule spans are used as noisy supervision for the rule extraction module. This preprocessing code is included with our code release.