Human Explanation-based Learning for Machine Comprehension

Qinyuan Ye [1] **Xiao Huang** [1] **Elizabeth Boschee** [2] **Xiang Ren** [1 2]

Abstract

Human annotators usually provide *only* the final labels in dataset collection process. Their rich knowledge and deductive power behind labeling decisions are not explicitly revealed in the dataset, while such information conveys the essence of human intelligence and is potentially generalizable to unseen cases. In this paper, we focus on the task of machine reading comprehension (MRC) and seek to obtain high-level human supervision in the form of semi-structured explanations that explicitly inform machines *why* an answer to a question is correct. Structured variables and rules are extracted from explanations to compose *neural module teachers*, which automatically annotate unlabeled data for training downstream MRC models. We show that our explanation-based annotation regime provides effective supervision, and is particularly efficient in low-resource scenarios. On the SQuAD dataset, our proposed method achieves 70.14% F1 score with 26 explanations, comparable to plain supervised learning using 1,100 labeled instances, yielding a 12x speed up[2].

1. Introduction

Recent advances in neural sequence learning and pre-trained language models yield human-level performance on several natural language processing tasks (Lan et al., 2019; Raffel et al., 2019). However, state-of-the-art results still heavily rely on large-scale human-annotated corpora, the collection of which involves repetitive manual labour and is often time-consuming. This leads to a large gap between methods in the research settings and practical use cases, as large amounts of annotated data rarely exist for a new task or a low-resource domain (Linzen, 2020). To reduce this dependency on annotation efforts, we seek to improve the efficiency in *obtaining* and *applying* human supervision.

One strength of human cognition is to generalize from relatively *few* examples; shown only a few instances of a problem and solution, humans often deduce patterns more readily than a machine, typically by bringing to bear a wealth of background information about what "really matters" in each example (DeJong & Mooney, 1986; Goldwasser & Roth, 2014; Lake et al., 2019). In contrast, traditional instance-level annotation may provide repeated information that are not best-suited for generalization. Based on these observations, we propose to collect human "deduction rules" in the form of semi-structured explanations and leverage them as high-level supervision for end task.

In this paper we focus on the machine reading comprehension (MRC) task, where the system is given a query and is asked to identify an answer span from a particular paragraph. Previous work soliciting explanations as part of the annotation process has been limited to classification tasks (Hancock et al., 2018; Wang et al., 2020). However, MRC is a more challenging target, since (1) it lacks explicit anchor words (*e.g.*, subject and object as in relation extraction); (2) has no pre-defined set of labels to choose from; and (3) the coverage for each explanation is even sparser.

Reference Instance
Q: When was Queen Victoria's *funeral* **held**?
C: Her *funeral* was **held** on Saturday, 2 February, in St George's Chapel, Windsor Castle, and after two days of lying-in-state ...
A: Saturday, 2 February
Semi-structured Explanation
X is "*funeral*". Y is "**held**". In the question X is within 4 words after "when was" and Y is directly after X. "on" is directly before the answer. Y is within 2 words before the answer. X is **within 3 words** left of Y. The question starts with "when", so the answer should be a date.
Strictly-matched Instance
Q: When was *independence* **declared**?
C: ... *Independence* was **declared** on 24 September 1973.
A: 24 September 1973
Softly-matched Instance
Q: When was *Brazelton* **killed**?
C: ... *Brazelton* was eventually tracked down and **killed** on Monday August 19, 1878, in a mesquite bosque ...
A: Monday August 19, 1878 (Confidence $z = 93.75\%$)
Note: X is 5 words left of Y, slightly violating "**within 3 words**".

Table 1. **Key elements in proposed work.** Semi-structured explanations characterize *why* an answer is correct and summarize the human's deductive process. Strictly- and softly-matched instances are automatically generated from explanations and provide supervision for training MRC models.

[1]Department of Computer Science, University of Southern California [2]Information Science Institute, University of Southern California. Correspondence to: Xiang Ren <xiangren@usc.edu>.

[2]Our code and data can be found at https://github.com/INK-USC/nl-explanation

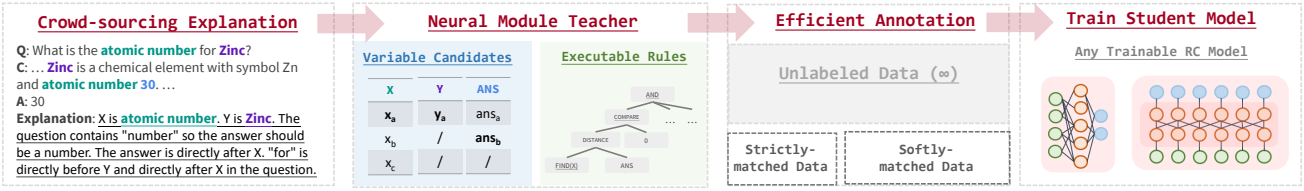

Figure 1. **Overview of proposed work.** We first collect a small set of semi-structured explanations, from which we extract key information such as variables and rules. These structured results are formulated into programs called neural module teachers (NMTeachers), which we use to curate supervision for training reading comprehension models.

To tackle these challenges, we propose the concept of a Neural Module Teacher (NMTeacher) – an executable *program* constructed from human-provided, semi-structured explanations that is (1) dynamically composed of modules based on the explanation; (2) capable of taking sequential steps and combinatorial search; and (3) capable of "fuzzy matching" using softened constraints. Fig. 1 shows an overview of our approach. We first use a Combinatory Categorial Grammar parser (Zettlemoyer & Collins, 2012) to turn explanations into structured *variables* and *rules* (Sec. 3.2). A neural module teacher is constructed with basic learnable modules (Sec. 3.1) based on parsing results and functions as a dedicated model for the specific type of question described in the explanation (Sec. 3.3). All neural module teachers act together and identify strictly- and softly-matched instances from an unlabeled corpus, which are used to train a "student" MRC model (Sec. 4.2). It is important to note that while this work is tied to the particular task of MRC, we believe it can be extended to a wide range of machine learning tasks.

We evaluated our approach on two datasets in extractive MRC setting: SQuAD v1.1 (Rajpurkar et al., 2016) and Natural Questions (Kwiatkowski et al., 2019). Experimental results highlight the efficiency of the proposed approach in extremely low-resource scenarios. Using 26 explanations gathered in 65 minutes, NMTeacher achieves 56.74% exact match and 70.14% F1 score on the SQuAD dataset, while the performance is 9.71% and 16.37% with traditional annotation obtained within the same amount of time. Moreover, our analysis shows that explanations continue to improve performance when a medium-sized manually-annotated dataset is readily available.

2. Problem Formulation

Our goal is to train an extractive MRC model $\mathbb{F}$, which takes as input a tuple (q, c) of question q and (document) context c, and extracts an answer span a within the context c. We are interested in gathering efficient and high-level human supervision for this task, and thus we assume a situation where a large set $\mathcal{S}$ of (q, c) pairs (without answer annotation) already exists, but we are limited to annotate only a small subset $\mathcal{S}_o$ (< 200 instances) of $\mathcal{S}$.

Overview and Notations. We provide an overview of our proposed method in Fig. 1. First, we collect an answer a_i and an explanation e_i for each (q_i, c_i) instance in $\mathcal{S}_o$, resulting in an updated $\mathcal{S}_o = \{(q, c, a, e)\}$. A neural module teacher $\mathbb{G}_i$ will be constructed from each explanation e_i, enabling it to answer questions similar to (q_i, c_i). All neural module teachers acting together can be viewed as an ensemble teacher $\mathbb{G}$. We then apply $\mathbb{G}$ to those unannotated (q, c) pairs in $\mathcal{S}$, getting $\mathcal{S}_a = \{(q, c, a)\}$, a strictly-labeled dataset that $\mathbb{G}$ can directly answer. The remaining unmatched instances are denoted as $\mathcal{S}_u = \{(q, c)\}$. After softening the constraints in each $\mathbb{G}_i$, we get a noisily-labeled dataset $\mathcal{S}_p = \{(q, c, a, z)\}$ from $\mathcal{S}_u$, where z is a confidence score given by $\mathbb{G}$. Notably, we will refer to the (q_i, c_i, a_i) part in $(q_i, c_i, a_i, e_i) \in \mathcal{S}_o$ as the "reference instance" for explanation e_i, as we will frequently check this (q_i, c_i, a_i) "for reference" when we apply $\mathbb{G}_i$ to unannotated instances.

$\mathcal{S}_a$ and $\mathcal{S}_p$ are significantly larger in size than $\mathcal{S}_o$ and thus provide more sufficient supervision. We use $\mathcal{S}_a$ and $\mathcal{S}_p$ to train a downstream student MRC model $\mathbb{F}$. We denote this method as NMTeacher-DA. We further explore variants such as (1) leveraging $\mathcal{S}_u$ with semi-supervised methods; and (2) joint training of $\mathbb{G}$ and $\mathbb{F}$. We construct our final model NMTeacher-Joint by incorporating these variants. Note that our approach is model-agnostic, so that $\mathbb{F}$ can take any trainable form.

3. Neural Module Teacher

A neural module teacher (NMTeacher) acts as a *program* that tries to answer questions following an explanation. In this section, we introduce the basic modules used for rule execution (Sec. 3.1), discuss how variables and rules are obtained from explanations (Sec. 3.2), and present how a neural module teacher derives answers (Sec. 3.3).

3.1. Atomic Modules

We define four types of atomic modules that can be composed to create neural module teachers: FILL, FIND, COMPARE and LOGIC. Each can support strict and softened matching criteria as a part of generating training instances for downstream use. We summarize their usage in Table 2 and introduce them in detail respectively in the following.

FILL Module: $(s_{ref}, p_{ref}, s) \rightarrow p$ **Description:** Select the span p in a given sentence s that plays the same syntactic role of span p_{ref} in sentence s_{ref}. **Example:** s_{ref} = How is packet switching characterized? p_{ref} = [2,3] (packet switching) s = How is hunting regulated? $\rightarrow p$ = [2,2] (hunting)
FIND Module: $(q_{ref}, p_{ref}, s) \rightarrow p$ **Description:** Find the span p in a context sentence s that refers to the span p_{ref} in the question q_{ref}. **Example:** q_{ref} = How is a promoter sequence recognized? p_{ref} = [2,4] (a promoter sequence) s = The promoter is recognized and bound by ... $\rightarrow p$ = [1,1] (promoter)
COMPARE Module: $(d_0, d_1) \rightarrow p$ **Description:** Softly evaluate the statement $d_1 \leq d_0$. **Example:** $d_0 = 0, d_1 = 1 \rightarrow p = 0.75$; $d_0 = 4, d_1 = 2 \rightarrow p = 1$
LOGICAND Module: $(p_1, p_2) \rightarrow p$ **Description:** Perform soft logic AND to two scalar probabilities. **Example:** $p_1 = 0.9, p_2 = 0.8 \rightarrow p = 0.7$; $p_1 = 1, p_2 = 1 \rightarrow p = 1$

Table 2. **Summary of atomic modules used in rule execution.** Rules constructed from explanations internally call these modules to fulfill complex functionalities. For example, LEFT(X, Y) is transformed to COMPARE(DISTANCE(FIND(X), FIND(Y)), 0)

FILL. When humans encounter a new question, they can detect structural similarities to previous questions they have seen. For example, humans will note that *How is hunting regulated?* is structurally similar to *How is packet switching characterized?*, enabling them to infer that answers to both might have a similar structure (*e.g.*, *by doing sth...*). To mimic this human intuition, we design a FILL module: given a sentence s_{ref} and a span of interest p_{ref}, FILL will predict analogous spans p in a new sentence s.

The *strict* version of FILL outputs spans p whose named entity type, dependency parse structure, or constituent parse structure[1] matches p_{ref}. We encourage over-generation since the rule execution step later on will verify each output. When strict matching yields nothing, we employ *softened* matching techniques. We first produce a contextualized phrase representation $\mathbf{e}'$ for p_{ref}. We then rank each candidate constituent p in sentence s according to the similarity between $\mathbf{e}'$ and an analogous phrase representation $\mathbf{e}$ for p. We return the top k constituents along with their score.

To generate phrase representations, we first encode the sentence with a BERT-base model (Devlin et al., 2019) and get representations $[\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_m]$ for each token. We then apply pooling over all tokens in span p to get the phrase representation $\mathbf{e}$. We considered both mean pooling and attentive pooling (Bahdanau et al., 2014). The similarity score between $\mathbf{e}$ and $\mathbf{e}'$ can be calculated using either cosine similarity, or learned bilinear similarity, *i.e.* $\text{Sim}(\mathbf{e}, \mathbf{e}') = \tanh(\mathbf{e}\mathbf{A}\mathbf{e}' + b)$, where $\mathbf{A}$ is a learnable matrix. We discuss pre-training and design choices for softened FILL module in Sec. 4.1.

[1]Identified using spaCy (https://spacy.io/)

FIND. The FILL module finds a span p that plays the same *role* as p_{ref} in its containing sentence. In contrast, FIND looks for a span p that has the same *meaning* as p_{ref}. For instance, if a query mentions *the explosion*, we might want to identify *exploded* as its counterpart in the paragraph being searched for an answer. This module is similar to the find module in Jiang & Bansal (2019) in its motivation, while we design ours to be a ranking-based module with discrete boundaries, so that the output fits in the search procedure in Sec. 3.3.

The strict version of FIND module directly looks for exact matches of the key p_{ref}. To account for synonyms, coreference, and morphological/spelling variation, we also build a softened version using the same model structure as the FILL module. We discuss the design choices and training for the softened FIND module in Sec. 4.1.

COMPARE. In our annotation guidelines, we encourage annotators to describe the relative location of spans in their explanations, *e.g.*, *X is within 3 words after Y*. The COMPARE module executes such distance comparisons. The strict version requires the condition to be met exactly: $P(d_1 \leq d_0) = 1$ when $d_1 \leq d_0$ and 0 otherwise. In the softened version, we attempt instead to indicate how close $d_1 \leq d_0$ is to being true:

$$P(d_1 \leq d_0) = \begin{cases} 1 & d_1 \leq d_0; \\ \max(1 - \frac{1}{4}(\frac{d_1 - d_0}{|d_0|+1})^2, 0) & d_1 > d_0. \end{cases} \quad (1)$$

As an example, $P(1 \leq 0) = 0.75$ (a near miss) but $P(5 \leq 0) = 0$ (due to the max in (1)).

LOGIC. The logic operations "and" and "or" often appear in explanations. A single explanation may also contain multiple sentences, requiring a logical AND to aggregate them. In the strict version of LOGIC, only boolean outputs of True (1) and False (0) are allowed. In the softened version, we use soft logic to aggregate two probabilities, *i.e.*, $\text{AND}(p_1, p_2) = \max(p_1 + p_2 - 1, 0)$ and $\text{OR}(p_1, p_2) = \min(p_1 + p_2, 1)$.

3.2. Parsing Explanations to Executable Rules

When soliciting explanations, we encourage annotators to think of each explanation as a collection of **variables** and **rules**. This framing allows us to effectively transform these explanations into executable forms. We formally define the terms here:

Variables are phrases that may be substituted in a question or answer when generalizing to unseen instances. In Table 1, underlined phrases are all considered variables. Annotators are guided to mark these spans explicitly, *e.g.*, *X is funeral*.

Explanation e**:** The answer is directly after X.
Parse p_j**:** @Is(Answer, @Direct(@Right(X)))
Execution f_j**:** COMPARE(DISTANCE(Ans,FIND(X)),0)
Explanation e**:** The answer is within 3 words before Z and within 4 words after Y.
Parse p_j**:** @Is(Answer,@And(@LessThan(@Left(Z), 3), @LessThan(@Right(Y, 4)))
Execution f_j**:** AND(COMPARE(DISTANCE(FIND(Z),Ans),3), COMPARE(DISTANCE(Ans,FIND(Y)),4))

Table 3. **Rules in three equivalent forms: explanation, parse and underlying execution.** Semi-structured explanations are first parsed and later transformed to executable functions. The execution form is composed of atomic modules (Sec. 3.1).

Y is held. X is within 5 words of Y. Variables are closely related to the design of the FILL module since FILL aims to propose potential assignments to these variables given unseen instances.

Rules are statements that describe characteristics of variables and relationships between them. When all variables in a rule are assigned, execution of a rule will output either True or False (strict) or a score between 0 and 1 (softened). Following previous work (Srivastava et al., 2017; Wang et al., 2020), we first use a Combinatory Categorial Grammar (CCG) based semantic parser $\mathbb{P}$ (Zettlemoyer & Collins, 2012) to transform explanations into logical forms (e.g., from e to p_j in Table 3). We build a domain-specific lexicon for common expressions used in explanations. We then implement the operation for each supported predicate (*e.g.*, "@Is", "@Direct", "@Left"), which may internally call atomic modules described in Sec 3.1. These predicate implementations, together with the inherent λ-calculus hierarchy from CCG, will yield the final executable function f_j as shown in Table 3.

3.3. Extracting Answer Spans

Rules introduced in Sec 3.2 can be executed to *verify* whether variable assignments are correct. In other words, given a (q, c, a) triple, executing all rules will give a boolean value (strict) or a confidence score (softened) indicating the triple's correctness. To actively *output* an answer, we need to re-formulate the problem so that each neural module teacher $\mathbb{G}_i$ takes (q, c) as input; and gives an answer span a and confidence score z as output. To this end, we formulate the task of extracting the best answer span into a combinatorial search problem, *i.e.*, searching for the best combination of variable assignments (including the answer).

To apply explanation e_i to a new question, candidates for each variable are first proposed by the FILL module. We then look for the best combination of variable assignments (achieving highest confidence) when evaluated using the rules generated from e_i. As a minimal example, if FILL proposes $\{x_1, x_2\}$ as potential assignments to variable X, and $\{a_1, a_2\}$ to ANS, we evaluate the four possible combinations $\{(x_1, a_1), (x_2, a_1), (x_1, a_2), (x_2, a_2)\}$ by applying e_i and select the one combination achieving highest confidence score. As the number of combinations expands significantly with the number of variables and their candidates, we solve this problem with beam search, progressively filling each variable and in each step keeping the most promising combinations (see Figure 6 and Algorithm 2 in appendix for more details). By doing so we have completed our construction of neural module teacher $\mathbb{G}_i$ from one semi-structured explanation e_i. We use $\mathbb{G}_i(q, c) = (a, z)$ to denote that given question q and context c, neural module teacher $\mathbb{G}_i$ identifies the answer span a with a confidence score z. Multiple neural module teachers $\mathbb{G}_i$ may ensemble into $\mathbb{G}$ by listing all answer spans outputted by each $\mathbb{G}_i$ and selecting the one with the highest z.

Algorithm 1 Learning with Explanations

Input: Tiny Dataset $\mathcal{S}_o = \{(q, c)\}$, Large Unlabeled Dataset $\mathcal{S} = \{(q, c)\}$, Confidence Threshold t
Output: MRC Model $\mathbb{F} : (q, c) \rightarrow a$

Collect Answers and Explanations for $\mathcal{S}_o$: $\mathcal{S}_o \leftarrow \{q, c, a, e\}$
// Construct Neural Module Teachers
$\mathbb{G} \leftarrow \varnothing$
for $(q_i, c_i, a_i) \in \mathcal{S}_o$ **do**
  Parse e_i and construct neural module teacher $\mathbb{G}_i$
  if $\mathbb{G}_i(q_i, c_i) = (a_i, 1.0)$ **then**
    $\mathbb{G} = \mathbb{G} \cup \{\mathbb{G}_i\}$ // $\mathbb{G}_i$ is validated
// Generate pseudo labels for $\mathcal{S}$
$\mathcal{S}_a \leftarrow \varnothing, \mathcal{S}_p \leftarrow \varnothing$
for $(q, c) \in \mathcal{S}$ **do**
  $(a, z) = \mathbb{G}(q, c)$ // z is confidence score
  if $z = 1$ **then**
    $\mathcal{S}_a \leftarrow \mathcal{S}_a \cup \{(q, c, a)\}$ // Strict Match
  else
    $\mathcal{S}_u \leftarrow \mathcal{S}_u \cup \{(q, c)\}$ // Unlabeled
    if $z > t$ **then**
      $\mathcal{S}_p \leftarrow \mathcal{S}_p \cup \{(q, c, a, z)\}$ // Softened Match
// Train Downstream MRC Model $\mathbb{F}$
$\mathbb{F} \leftarrow$Train$(\mathcal{S}_a, \mathcal{S}_p, \mathcal{S}_u)$
return $\mathbb{F}$

4. Learning to Augment with NMTeacher

4.1. Pre-training of Fill and Find Module

The softened FILL module is pre-trained with pairs of (positive) matches (q_{ref}, s_{ref}, q, s) from strictly-matching results $\mathcal{S}_a$, including 99153 questions and 55202 contexts, divided into 70% train, 10% dev and 20% test datasets. We use random constituents in the sentence as negative training examples. For the FILL module, we evaluated various model designs described in section 3.1 and choose to use attentive pooling and bilinear similarity.

The softened FIND module assesses semantic similarity of phrases. We tried various datasets as proxies for pre-training this ability, including coreference resolution results on SQuAD corpus (produced by Stanford CoreNLP (Man-

ning et al., 2014)) and paraphrase dataset (PPDB (Pavlick et al., 2015)). We manually evaluated FIND module performance with $\mathcal{S}_o$, and we observe that using mean pooling and cosine similarity without any pre-training yields the best performance. We conjecture this may be caused by data bias (the proxy training data not aligning with the purpose of the module). To this end, we use untrained BERT-base as our FIND module to capture semantic similarities. We leave our manual evaluation results in Appendix B.

4.2. Joint Learning with MRC Model

Our learning framework (Algorithm 1) uses our ensemble neural module teacher $\mathbb{G}$ to answer each (q, c) instance in $\mathcal{S}$, resulting in three sets of data instances: a strictly-matched set $\mathcal{S}_a$, a softly-matched set $\mathcal{S}_p$ and an unlabeled set $\mathcal{S}_u$. We use these three sets to jointly learn our downstream MRC model and NMTeacher, as described below.

Learning from Strictly-matched Data $\mathcal{S}_a$. We start by simply treating $\mathcal{S}_a$ as a labeled dataset, and first train the downstream MRC model $\mathbb{F}$ with traditional supervised learning. We compare different MRC models in our experiments. For simplicity, we denote $\texttt{MRC\_Loss}(\mathcal{B}^{(i)})$ as the loss term defined in these MRC models for the i-th instance in batch $\mathcal{B}$. In each step, we sample a batch $\mathcal{B}_a$ from $\mathcal{S}_a$ and update the model with loss term $\mathcal{L}(\mathcal{B}_a)$:

$$\mathcal{L}(\mathcal{B}_a) = \sum_{i=1}^{|\mathcal{B}_a|} \frac{1}{|\mathcal{B}_a|} \cdot \texttt{MRC\_Loss}(\mathcal{B}_a^{(i)}). \tag{2}$$

Learning from Softly-matched Data $\mathcal{S}_p$. The softly-matched set $\mathcal{S}_p$ is significantly larger in size (than $\mathcal{S}_a$) and may contain useful information for training $\mathbb{F}$. We blend in supervision from $\mathcal{S}_p$ by adding a weighted loss term to the original loss $\mathcal{L}(\mathcal{B}_a)$. That is, we simultaneously sample a batch $\mathcal{B}_a$ from $\mathcal{S}_a$ and a batch $\mathcal{B}_p$ from $\mathcal{S}_p$. The loss term for $\mathcal{B}_p$ is weighted and normalized by the confidence score z from NMTeacher $\mathbb{G}$,

$$w_i = \frac{\exp(\theta_t z_i)}{\sum_{j=1}^{|\mathcal{B}_p|} \exp(\theta_t z_j)}, \tag{3}$$

$$\mathcal{L}(\mathcal{B}_p) = \sum_{i=1}^{|\mathcal{B}_p|} w_i \cdot \texttt{MRC\_Loss}(\mathcal{B}_p^{(i)}), \tag{4}$$

where θ_t in Eq. 3 is a temperature that controls the normalization intensity. We then aggregate the loss terms from $\mathcal{S}_p$ and $\mathcal{S}_a$ with coefficient β, *i.e.*, $\mathcal{L}_{ap} = \mathcal{L}(\mathcal{B}_a) + \beta\mathcal{L}(\mathcal{B}_p)$. We denote the method up to this step as NMTeacher-DA.

Learning from Unlabeled Data $\mathcal{S}_u$. We further learn from unlabeled data in $\mathcal{S}_u$ by integrating existing semi-supervised methods. In brief, pseudo labeling (PL) samples a batch $\mathcal{B}_u$ from $\mathcal{S}_u$, annotates it with the current MRC model $\mathbb{F}$, and calculates the loss term on this pseudo-labeled batch $\mathcal{B}_u$. The overall loss $\mathcal{L}$ term thus becomes $\mathcal{L}_{au} = \mathcal{L}(\mathcal{B}_a) + \beta\mathcal{L}(\mathcal{B}_u)$. To mix in supervision from unlabeled data, we formulate a $r+1$ rotation between sampling unlabeled batch $\mathcal{B}_u$ and softly-matched batch $\mathcal{B}_p$; we update MRC model $\mathbb{F}$ for r steps using the semi-supervised method and loss term $\mathcal{L}_{au}$, and then update the model for one step using softly-matched data and the loss term $\mathcal{L}_{ap}$.

Joint Training. Instance weight w_i (Eq. 3) for each softly-labeled instance in batch $\mathcal{B}_p$ is calculated with NMTeacher $\mathbb{G}$, so we further allow gradient backpropagation to trainable FILL and FIND modules in $\mathbb{G}$ when optimizing loss term $\mathcal{L}_{au}$. We fix $\mathbb{G}$ at first and allow joint training after training on $\mathbb{F}$ converges. This helps form consensus between NMTeacher $\mathbb{G}$ and the learned downstream MRC model $\mathbb{F}$, which we believe is helpful in denoising and refining the final MRC model. We denote this final method as NMTeacher-Joint.

5. Experiments

5.1. Experiment Setup

Datasets. (1) **SQuAD v1.1** (Rajpurkar et al., 2016) contains over 10k crowd-sourced MRC instances. All questions are answerable. (2) **Natural Questions (NQ)** (Kwiatkowski et al., 2019) contains questions from Google search queries, paired with related Wikipedia articles. For consistency between datasets, we follow the setting where "the long answer is given, and a short answer is known to exist". We further discard instances whose (1) long answer is not free-form text (*e.g.*, table, list); or (2) short answer contains multiple short spans.

Evaluation. Use of the official SQuAD and NQ test sets is restricted, therefore we construct our own dev and test sets by splitting the official dev sets in half.[2] Hyper-parameters and the best checkpoint are selected on the dev set. We use the SQuAD official evaluation script and report Exact Match (EM) and F1 score on both the dev set (in appendix) and test set (in Sec 5.2). We report 3-run average and standard deviation in all our experiments.

MRC Models. Importantly, our approach is model-agnostic. We test our framework using the following three models for downstream reading comprehension. (1) **BiDAF** (Seo et al., 2016), which adopts hierarchical architecture and attention mechanism to model question-context interactions; (2) **BERT** (Devlin et al., 2019), a pre-trained language model with an additional output layer for MRC[3]; and (3)

[2] SQuAD: 5537 dev / 5033 test. NQ: 1252 dev / 1252 test.

[3] We use BERT-l as a short hand for BERT-large and BERT-b for BERT-base in following analysis.

#Explanations ($\lvert\mathcal{S}_a\rvert$, $\lvert\mathcal{S}_p\rvert$)	13 (131, 314)		26 (424, 1048)		52 (766, 2329)	
	EM	F1	EM	F1	EM	F1
BiDAF ($\mathcal{S}_a$)	3.66 ± 0.92	7.80 ± 0.84	5.49 ± 0.50	9.91 ± 0.34	8.21 ± 0.25	14.15 ± 0.40
+ NMTeacher-DA ($\mathcal{S}_p$)	5.15 ± 0.45	8.51 ± 0.22	6.65 ± 0.34	11.46 ± 0.49	12.63 ± 0.86	19.99 ± 1.06
BERT-base ($\mathcal{S}_a$)	10.52 ± 1.57	17.88 ± 2.09	19.90 ± 1.53	30.42 ± 1.53	28.84 ± 1.69	39.26 ± 2.12
+ NMTeacher-DA ($\mathcal{S}_p$)	13.80 ± 1.29	23.39 ± 1.43	22.30 ± 2.78	32.96 ± 5.00	30.74 ± 2.48	41.28 ± 3.14
BERT-large ($\mathcal{S}_a$)	13.27 ± 1.09	21.11 ± 2.26	25.90 ± 2.55	38.35 ± 2.38	34.66 ± 0.65	47.32 ± 0.60
+ NMTeacher-DA ($\mathcal{S}_p$)	15.80 ± 1.64	27.45 ± 2.32	28.07 ± 2.27	41.95 ± 2.95	39.05 ± 1.36	51.65 ± 2.08
+ Self Training ($\mathcal{S}_u$)	15.25 ± 2.49	23.13 ± 2.84	30.43 ± 6.30	40.80 ± 4.53	43.55 ± 3.39	54.62 ± 4.40
+ Mean Teacher ($\mathcal{S}_u$)	11.84 ± 2.36	19.62 ± 2.37	32.80 ± 5.72	45.50 ± 4.61	41.86 ± 7.22	54.74 ± 5.80
+ Pseudo Labeling ($\mathcal{S}_u$)	14.82 ± 1.70	21.67 ± 2.96	38.10 ± 5.62	50.62 ± 7.30	50.45 ± 2.11	61.82 ± 1.32
+ NMTeacher-Joint ($\mathcal{S}_p + \mathcal{S}_u$)	34.80 ± 14.16	44.00 ± 17.74	**56.74 ± 1.27**	70.14 ± 2.58	**58.11 ± 0.95**	70.67 ± 1.58
ALBERT-base ($\mathcal{S}_a$)	30.12 ± 1.00	42.95 ± 1.65	39.24 ± 1.80	53.40 ± 2.87	44.57 ± 1.90	58.09 ± 0.59
+ NMTeacher-DA ($\mathcal{S}_p$)	34.31 ± 1.23	46.59 ± 1.16	40.79 ± 0.78	55.22 ± 0.29	46.55 ± 1.04	59.80 ± 0.64
+ Self Training ($\mathcal{S}_u$)	35.45 ± 3.58	45.27 ± 3.71	46.21 ± 3.46	58.20 ± 4.04	47.08 ± 3.70	60.57 ± 4.11
+ Mean Teacher ($\mathcal{S}_u$)	29.35 ± 1.79	41.73 ± 1.07	40.92 ± 2.05	55.17 ± 2.36	52.16 ± 0.66	65.83 ± 1.52
+ Pseudo Labeling ($\mathcal{S}_u$)	27.35 ± 2.66	39.95 ± 4.24	38.56 ± 2.81	51.77 ± 2.53	43.76 ± 1.88	56.69 ± 2.50
+ NMTeacher-Joint ($\mathcal{S}_p + \mathcal{S}_u$)	**40.67 ± 5.48**	**52.49 ± 4.74**	54.88 ± 3.16	**70.21 ± 3.21**	57.69 ± 0.77	**71.75 ± 0.48**

Table 4. **Performance comparison on SQuAD using 13/26/52 explanations.** $\mathcal{S}_a$ is the set of strictly matched instances annotated by NMTeacher. $\mathcal{S}_p$ is the set of softly matched instances by using softened modules in rule execution. $\mathcal{S}_p$ constantly brings improvements over model trained solely on $\mathcal{S}_a$, showing that the usage of softly-matched but noisy data is beneficial. Such improvement is most significant in extreme low-resource cases with 13 explanations. Best performance is achieved when semi-supervised learning on unlabeled data $\mathcal{S}_u$ and joint training of NMTeacher and MRC model are enabled (NMTeacher-Joint).

Statistics / Dataset	SQuAD	NQ
# Collected raw explanations	2,065	1,220
# Accepted explanations	570	343
# Parsable explanations	163	109
# Validated explanations	130	89
Average # sentences per explanation	4.31	4.51
Average # tokens per explanation	38.87	41.28
Average # variables per explanation	1.96	1.47

Table 5. **Statistics of the collected explanations.**

ALBERT (Lan et al., 2019), a top-performing model on the SQuAD leaderboard.

Semi-supervised Methods. We compare and enhance NMTeacher with the following semi-supervised methods: (1) **Self Training (ST)** (Rosenberg et al., 2005) iteratively annotates unlabeled instances with maximal confidence in each epoch; (2) **Pseudo Labeling (PL)** (Lee, 2013) trains a weak model on labeled data first and annotates unlabeled batches as supervision. (3) **Mean Teacher (MT)** (Tarvainen & Valpola, 2017) introduces consistency loss between a student model and a teacher model (the exponential moving average of student models from previous steps).

Explanation Collection. Table 5 provides statistics on the explanations we collected for this effort. We refer readers to Appendix E for more details, including our crowdsourcing interface and guidelines. On average, annotators spend 43 seconds to annotate an answer and 151 seconds to annotate both an explanation and an answer (3.5x slower compared to annotating answer only).

5.2. Performance Comparison

Main Results. Tables 4 and 6 show results of different MRC models, with different numbers of explanations used. The baseline for each model uses as training the strictly-matched instances ($\mathcal{S}_a$) generated using the explanations. For all models, performance improves when we include the softly-matched instances ($\mathcal{S}_p$). We show in Fig. 2 that this pattern largely continues even as we further increase the number of explanations, showing that noisy labels are of highest value in low-resource settings but still continue to provide value as training sizes increase. In most cases, performance improves further when trained with semi-supervised learning and $\mathcal{S}_u$. Finally, performance is best when we make full use of $\mathcal{S}_a$, $\mathcal{S}_p$ and $\mathcal{S}_u$, and jointly train $\mathbb{F}$ and $\mathbb{G}$ (NMTeacher-Joint).

Efficiency Study. We demonstrate NMTeacher's efficiency by controlling annotation time. Given a fixed amount of time t, we denote $\mathcal{S}_r^{(t)}$ as plain answers that could be collected in t; $\mathcal{S}_a^{(t)}$ and $\mathcal{S}_p^{(t)}$ as strictly and softly matched data generated by answers + explanations collected in t. We train a BERT-l MRC model in the following settings: (1) Supervised learning with $\mathcal{S}_r^{(t)}$; (2) NMTeacher-DA with $\mathcal{S}_a^{(t)}$ and $\mathcal{S}_p^{(t)}$; (3) NMTeacher-Joint. Fig. 3 shows that NMTeacher significantly improves performance over the baseline when annotation time is constant. Additionally, we found that the 70.14% F1 score achieved with 26 explanations, requires 1,100 annotated examples if put in supervised learning setting. This gives a 12x annotation speed up.

#Explanations ($\lvert\mathcal{S}_a\rvert$, $\lvert\mathcal{S}_p\rvert$)	18 (98, 539)		36 (107, 647)		54 (273, 1047)	
	EM	F1	EM	F1	EM	F1
BERT-l ($\mathcal{S}_a$)	11.63 ± 1.52	20.86 ± 1.78	15.26 ± 0.55	24.89 ± 1.47	14.24 ± 0.74	24.85 ± 1.77
+ NMTeacher-DA ($\mathcal{S}_p$)	17.47 ± 0.76	28.30 ± 0.42	20.77 ± 2.04	31.86 ± 2.37	19.33 ± 2.44	31.56 ± 2.55
+ Self Training ($\mathcal{S}_u$)	15.92 ± 2.13	25.17 ± 0.65	18.42 ± 0.67	27.85 ± 0.46	17.49 ± 1.67	26.18 ± 0.55
+ Mean Teacher ($\mathcal{S}_u$)	14.67 ± 0.32	24.63 ± 0.57	17.94 ± 0.93	27.71 ± 0.98	17.63 ± 1.32	27.12 ± 1.24
+ Pseudo Labeling ($\mathcal{S}_u$)	17.86 ± 1.71	25.47 ± 0.36	20.18 ± 2.35	27.60 ± 2.40	16.56 ± 0.41	25.80 ± 0.66
+ NMTeacher-Joint ($\mathcal{S}_p + \mathcal{S}_u$)	17.36 ± 0.70	28.36 ± 1.09	23.22 ± 1.74	33.93 ± 2.16	24.04 ± 2.90	34.90 ± 2.65
ALBERT-b ($\mathcal{S}_a$)	19.62 ± 2.39	27.84 ± 2.89	21.78 ± 2.93	31.20 ± 3.46	21.19 ± 1.80	32.08 ± 1.48
+ NMTeacher-DA ($\mathcal{S}_p$)	21.17 ± 1.48	30.67 ± 2.47	25.93 ± 3.91	35.82 ± 3.73	23.16 ± 4.26	33.89 ± 3.59
+ Self Training ($\mathcal{S}_u$)	19.41 ± 1.31	28.04 ± 1.71	22.15 ± 2.50	31.09 ± 2.30	21.65 ± 2.92	31.08 ± 2.93
+ Mean Teacher ($\mathcal{S}_u$)	20.26 ± 0.65	29.25 ± 0.14	24.71 ± 3.38	33.66 ± 3.65	28.06 ± 2.48	37.91 ± 2.15
+ Pseudo Labeling ($\mathcal{S}_u$)	18.88 ± 1.98	27.28 ± 1.88	23.30 ± 2.67	31.96 ± 1.46	20.23 ± 1.43	30.62 ± 2.63
+ NMTeacher-Joint ($\mathcal{S}_p + \mathcal{S}_u$)	**24.12 ± 4.12**	**34.65 ± 5.03**	**30.56 ± 2.42**	**41.14 ± 3.10**	**29.45 ± 3.64**	**41.14 ± 3.14**

Table 6. **Performance comparison on NQ using 18/36/54 explanations.** Similar trends as in Table 4 can be observed.

No.	Training Supervision	EM	F1
(1)	$\mathcal{S}_a$	44.57 ± 1.90	58.09 ± 0.59
(2)	$\mathcal{S}_a$+$\mathcal{S}_p$	46.55 ± 1.90	59.80 ± 0.64
(3)	$\mathcal{S}_a^*$	52.14 ± 2.02	64.25 ± 1.89
(4)	$\mathcal{S}_a^*$+$\mathcal{S}_p^*$	59.67 ± 0.33	71.55 ± 0.34
(5)	$\mathcal{S}_r(\lvert\mathcal{S}_r\rvert = \lvert\mathcal{S}_a\rvert)$	59.15 ± 0.88	71.40 ± 0.61
(6)	$\mathcal{S}_r(\lvert\mathcal{S}_r\rvert = \lvert\mathcal{S}_a\rvert + \lvert\mathcal{S}_p\rvert)$	69.27 ± 0.30	80.09 ± 0.66

Table 7. **Analysis on Matching Quality.** $\mathcal{S}_a$ and $\mathcal{S}_p$ are obtained with 52 explanations. $\mathcal{S}_a^*$ denotes instances in $\mathcal{S}_a$ paired with gold-standard human annotations. $\mathcal{S}_r$ is randomly sampled from SQuAD with size controlled to be equal to $\lvert\mathcal{S}_a\rvert$ or $\lvert\mathcal{S}_a\rvert + \lvert\mathcal{S}_p\rvert$.

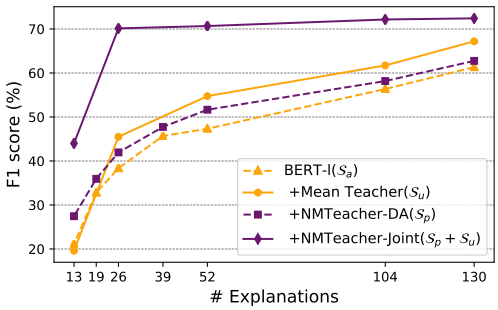

Figure 2. **Performance changes with respect to number of explanations on SQuAD.** Performance of the proposed method grow progressively with more explanations.

5.3. Performance Analysis

Matching Noise / Bias. Our proposed method hypothesizes new training examples, which may be noisy even when "strictly-matched". The matched instances may also be more similar than desired to the reference instances. To assess the impact of these two factors, we look at the strictly-matched set $\mathcal{S}_a$ and the softly-matched set $\mathcal{S}_p$ generated with 52 SQuAD explanations. We define $\mathcal{S}_a^*$ and $\mathcal{S}_p^*$, versions of these sets with human-annotated answers (*i.e.*, no noise). We then train an ALBERT-b model with supervision in the following six settings: (1) $\mathcal{S}_a$; (2) $\mathcal{S}_a$ and $\mathcal{S}_p$; (3) $\mathcal{S}_a^*$; (4) $\mathcal{S}_a^*$ and $\mathcal{S}_p^*$; (5) $\mathcal{S}_r$, a set of randomly sampled SQuAD training instances with size $\lvert\mathcal{S}_a\rvert$; (6) $\mathcal{S}_r$ of size $\lvert\mathcal{S}_a\rvert + \lvert\mathcal{S}_p\rvert$. Results are listed in Table 7. Comparing (1) and (3), we observe a 6.16% F1 gap caused by noise in strict matching; (2) and (4) show that the gap is further widened, since there are more noises in softly-matched data. Comparing (3) and (5), we see a 7.15% F1 gap mainly caused by bias in the instances selected by NMTeachers. We believe addressing these two issues will improve model performance, and we leave them as future work.

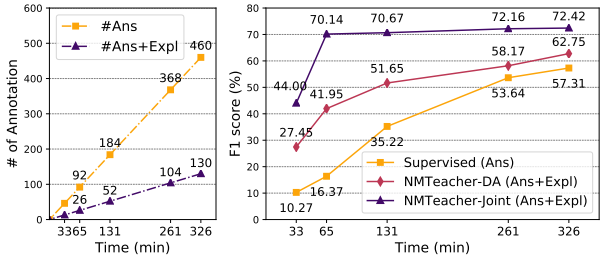

Figure 3. **Study on Annotation Efficiency.** We compare model performance when annotation time is held constant; NMTeacher-Joint consistently outperforms the baseline without explanations (*e.g.*, 70.14% vs. 16.37% F1 score with 65 minutes annotation). BERT-l is used as MRC model.

Medium Resource Scenarios. Going beyond low-resource cases, we show that NMTeacher could extend to medium-resource scenarios. We randomly sample $\{1000, 2000, 3000, 5000\}$ human-annotated instances from SQuAD as $\mathcal{S}_r$. We train a BERT-l MRC model using $\mathcal{S}_r$ along with $\mathcal{S}_a$, $\mathcal{S}_p$ generated with 52 explanations. We compare with training the MRC model with $\mathcal{S}_r$ only. Fig. 5 shows that when a medium-size $\mathcal{S}_r$ is readily available, augmenting it with NMTeacher is still beneficial. This could be particularly useful when a defect is observed in the trained model (*e.g.*, a certain type of question is answered poorly). A small set of explanations could be collected rapidly and used as "hot-fix".

Ablation Study on Modules. To evaluate the effect of the softened module execution, we progressively turn on the softened version of FIND, FILL and COMPARE in NMTeacher matching process, use matched data to train the downstream MRC model $\mathbb{F}$ in NMTeacher-DA setting, and report the final performance. The evaluation results are presented in Fig. 4. Results show that softening each

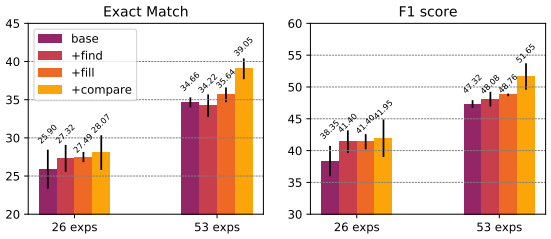

Figure 4. **Ablation study on atomic modules.** Fill, Find and compare modules are switched to softened mode consecutively. Rule softening in each module contributes to improve final MRC model performance.

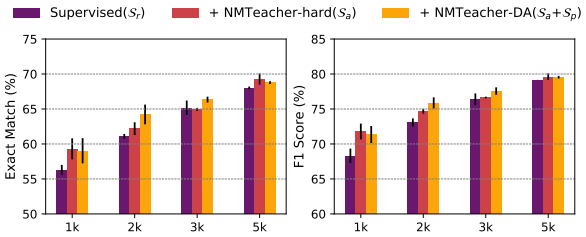

Figure 5. **Augmenting Labeled Instances with Explanations in medium-resource scenarios**. Please refer to Sec 5.3 "Medium Resource Scenarios" for in-depth analysis.

module contributes to performance improvement.

Statistics of strictly-matched instances. We list the "question heads" in $\mathcal{S}_a$ and found the top 8 to be: when did (22.08%); what year (8.51%); how many (8.1%); who was (7.27%); what did (6.43%); what percentage (5.39%); what does (5.26%); how long (4.35%). This observation demonstrates the explanations we collect cover a wide range of question types. Yet, the distribution of input data has far more aspects than question heads. Our current implementation and design may not explain complex questions that require multi-step reasoning abilities, and this may result in strong biases in $\mathcal{S}_a$ and $\mathcal{S}_p$.

To examine the labeling accuracy, we directly evaluate annotations obtained with the neural module teacher $\mathbb{G}$ against human annotations. On SQuAD with 52 explanations, 72.19% EM and 83.35% F1 is achieved on the 766 strictly-matches instances in $\mathcal{S}_a$. Noises in annotations generated with neural module teachers $\mathbb{G}$ also cause performance downgrade in the final model $\mathbb{F}$; and thus denoising matched instances will help improve performance. Joint training may partially resolve this by encouraging consensus between $\mathbb{G}$ and MRC model $\mathbb{F}$; meanwhile we encourage future research in this direction.

6. Related Work

Learning with Explanations. Srivastava et al. (2017) first propose to use explanations as *feature* functions in concept learning. Hancock et al. (2018) proposed BABBLELABBLE for training classifiers with explanations in data programming setting, which uses explanations to provide *labels* instead of features. Wang et al. (2020) proposed NEXT to improve generalization of explanations with softened rule execution. Both BABBLELABBLE and NEXT highlight annotation efficiency in low-resource settings. To the best of our knowledge, we are the first to study soliciting explanations for MRC, which is intrinsically more challenging than classification tasks in existing works.

Learning from Unlabeled data. A notable line of work focuses on enforcing consistency on unlabeled data by regularizing model predictions to be invariant to noise-augmented data (Xie et al., 2019; Yu et al., 2018). Consistency can also be enforced through temporal ensemble (Laine & Aila, 2016; Tarvainen & Valpola, 2017). Another line of work uses bootstrapping – first training a weak model with labeled data; then use model prediction on unlabeled data as supervision (Carlson et al., 2009; Yang et al., 2018a). Our proposed method is non-conflicting with semi-supervised strategies and we enhance NMTeacher with these strategies to achieve the best performance.

Neural Module Networks. Neural module networks (NMNs) are dynamically composed of individual modules of different capabilities. It was first proposed for VQA tasks (Andreas et al., 2016b;a; Hu et al., 2017). Recently in NLP community, reading comprehension requiring reasoning (Yang et al., 2018b; Dua et al., 2019; Amini et al., 2019) are proposed and widely studied. Recent works (Jiang & Bansal, 2019; Gupta et al., 2020) generally adopt a parser that gives a sequence of operations to derive the final answer. Our work differs in that (1) operations are constructed from explanations instead of questions; (2) NMTeacher provides supervision, instead of being used as final MRC model and trained in a fully-supervised manner. We limit our scope to SQuAD-style MRC tasks in this paper and leave other challenging tasks as future work.

7. Conclusion

We propose a novel framework that efficiently gathers supervision for training machine reading comprehension models. We begin with a small set of human-provided semi-structured explanations and compose NMTeachers to program and augment unlabeled data. NMTeachers are modularized functions where each module has a strict and softened form, enabling broader coverage from each explanation. Extensive experiments on two datasets and multiple MRC models demonstrate the effectiveness of our system in low-resource and medium-resource settings. Having achieved encouraging results for MRC, we look forward to extending this framework for more challenging tasks such as knowledge acquisition and multi-hop reasoning.

References

Amini, A., Gabriel, S., Lin, S., Koncel-Kedziorski, R., Choi, Y., and Hajishirzi, H. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of NAACL-HLT*, pp. 2357–2367, 2019.

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. Learning to compose neural networks for question answering. In *Proceedings of NAACL-HLT*, pp. 1545–1554, 2016a.

Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 39–48, 2016b.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Carlson, A., Gaffney, S., and Vasile, F. Learning a named entity tagger from gazetteers with the partial perceptron. In *AAAI Spring Symposium: Learning by Reading and Learning to Read*, 2009.

DeJong, G. and Mooney, R. Explanation-based learning: An alternative view. *Machine learning*, 1(2):145–176, 1986.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 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)*, pp. 4171–4186, 2019.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*, 2019.

Goldwasser, D. and Roth, D. Learning from natural instructions. *Machine learning*, 94(2):205–232, 2014.

Gupta, N., Lin, K., Roth, D., Singh, S., and Gardner, M. Neural module networks for reasoning over text. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SygWvAVFPr.

Hancock, B., Bringmann, M., Varma, P., Liang, P., Wang, S., and Ré, C. Training classifiers with natural language explanations. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2018, pp. 1884. NIH Public Access, 2018.

Hu, R., Andreas, J., Rohrbach, M., Darrell, T., and Saenko, K. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 804–813, 2017.

Jiang, Y. and Bansal, M. Self-assembling modular networks for interpretable multi-hop 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)*, pp. 4464–4474, 2019.

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

Laine, S. and Aila, T. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

Lake, B. M., Linzen, T., and Baroni, M. Human few-shot learning of compositional instructions. *arXiv preprint arXiv:1901.04587*, 2019.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Lee, D.-H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, pp. 2, 2013.

Linzen, T. How can we accelerate progress towards human-like linguistic generalization? *arXiv preprint arXiv:2005.00955*, 2020.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., and McClosky, D. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pp. 55–60, 2014.

Pavlick, E., Rastogi, P., Ganitkevitch, J., Van Durme, B., and Callison-Burch, C. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 425–430, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-2070. URL https://www.aclweb.org/anthology/P15-2070.

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

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

Rosenberg, C., Hebert, M., and Schneiderman, H. Semi-supervised self-training of object detection models. *WACV/MOTION*, 2, 2005.

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

Srivastava, S., Labutov, I., and Mitchell, T. Joint concept learning and semantic parsing from natural language explanations. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pp. 1527–1536, 2017.

Tarvainen, A. and Valpola, H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pp. 1195–1204, 2017.

Wang, Z., Qin, Y., Zhou, W., Yan, J., Ye, Q., Neves, L., Liu, Z., and Ren, X. Learning from explanations with neural execution tree. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJlUt0EYwS.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. Unsupervised data augmentation for consistency training. 2019.

Yang, Y., Chen, W., Li, Z., He, Z., and Zhang, M. Distantly supervised ner with partial annotation learning and reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2159–2169, 2018a.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018b.

Yu, A. W., Dohan, D., Luong, M.-T., Zhao, R., Chen, K., Norouzi, M., and Le, Q. V. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

Zettlemoyer, L. S. and Collins, M. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*, 2012.

Appendix:
Human Explanation-based Learning for Machine Comprehension

A. Case Study

Strictly-matched instances. Table 8 shows two examples of strictly-matched instances. In the first example, the explanation specified how to answer questions similar to "In what year did X (`sth.`) begin". Intuitively, the answer should be a year number right after "since", and the entity before "begin" should be a keyword. In the second example, questions following the pattern "when was X (`sth.`) Y (`done`)" are explained and the answer is typically a date after "on". Also, the verb "`done`" should be directly before "on" and the answer.

Softly-match Instances. Table 9 shows two examples of softly-matched instances. In the first example, the distance between Y and Z is three in the question, while the explanation specifies there should be less than two words between them. With COMPARE module, the correct answer is found with high confidence of 97.22%. In the second example, the explanation specifies Y to be an adjective phrase. With FILL module, a verb in the past tense, "purified", is also listed as a potential fit for variable Y, and this gives the correct answer "a secret lake" with a confidence score of 72.48%.

Reference Instance
Q: In what year did *Film Fest New Haven* begin?
C: ... The *Film Fest New Haven* has been held annually since 1995.
A: 1995
Semi-structured Explanation
X is "*Film Fest New Haven*". The question starts with "In what year", so the answer should be a year. "begin" is in the question. X is directly after "did" and directly before "begin" in the question. "since" is directly before the answer.
Strictly-matched Instance
Q: In what year did *the Music of the Night* begin?
C: ... Since 1992 *the Music of the Night* has been performed in the Royal Citadel by the 29 Commando Regiment and local performers to raise money for local and military charities. ...
A: 1992

Reference Instance
Q: When was Queen Victoria's *funeral* held?
C: Her *funeral* was held on Saturday, 2 February, in St George's Chapel, Windsor Castle, and after two days of lying-in-state ...
A: Saturday, 2 February
Semi-structured Explanation
X is "*funeral*". Y is "held". In the question X is within 4 words after "when was" and Y is directly after X. "on" is directly before the answer. Y is within 2 words before the answer. X is **within 3 words** left of Y. The question starts with "when", so the answer should be a date.
Strictly-matched Instance
Q: When was *independence* declared?
C: ... *Independence* was declared on 24 September 1973.
A: 24 September 1973

Table 8. **Examples of strictly-matched instances.**

Reference Instance
Q: Who did *Estonia* rebel against in 1343?
C: ... In 1343, the people of northern *Estonia* and Saaremaa rebel against German rule in the St. George's Night Uprising , which was put down by 1345. ...
A: German rule
Semi-structured Explanation
X is "*Estonia*". Y is "rebel against". Z is "1343". In the question, Y is directly after X and Z is within 2 words after Y. Z is a year. The answer directly follows Y. X is within 3 words before Y.
Softly-matched Instance
Q: *The Slavs* appeared on whose borders around the 6th century?
C: ... Around the 6th century, *Slavs* appeared on Byzantine borders in great numbers. ...
A: Byzantine borders (Confidence z = 97.22%)
Note
Z (the 6th century) is 3 words after Y (appeared on) in the question, which slightly breaks the constraint "Z is within 2 words after Y". This is captured by COMPARE module.

Reference Instance
Q: Where is *hydrogen* highly soluble?
C: ... *Hydrogen* is highly soluble in many rare earth and transition metals and is soluble in both nanocrystalline and amorphous metals. ...
A: many rare earth and transition metals
Semi-structured Explanation
X is "*hydrogen*". Y is "highly soluble". Y is directly after X and X is directly after "where is" in the question. X is within 5 words before Y. Y is within 2 words before the answer. "in" directly before the answer. "is" is between X and Y.
Softly-matched Instance
Q: Where is *the divinity herself* purified?
C: ... Afterwards the car, the vestments, and, if you like to believe it, *the divinity herself*, are purified in a secret lake. ...
A: a secret lake (Confidence z = 72.48%)
Note
In the reference instance, Y (highly soluble) is supposed to be an adjective phrase. In the new instance, FILL module suggested "purified" to be a promising candidate for variable Y.

Table 9. **Examples of softly-matched instances.**

B. Additional Performance Analysis

Performance of Fill and Find module The FILL module is evaluated on the test split of hard-matched question pairs and context pairs, as described in Sec. 4.1. The FIND module is evaluated through manual inspection on model's predictions on instances in $\mathcal{S}_o$. For each sentence in the test set, we enumerate all possible constituents, let the model rank these spans. We take top-n ($n = 1, 3, 5, 10$ for FILL module and $n = 1$ for FIND module) spans as output. We use recall (at n) $r_n = \frac{p}{q}$ as metric for evaluation, where p is the number of correct spans found in top-n outputs and q is the number of all correct spans. Evaluation results for Fill and Find module are shown in Table 10. As n gets large, the top-n outputs from the FILL module are able to identify most of the correct spans.

Recall@n (%)	Top-1	Top-3	Top-5	Top-10
Fill (Questions)	68.50	88.01	94.66	98.93
Fill (Contexts)	95.64	97.45	98.22	98.73
Find	41.00	-	-	-

Table 10. Evaluation on Fill and Find module. We evaluate Fill on the test split (described in Sec. 4.1) and Find on collected explanations and their reference instances.

Algorithm 2 Beam Search for NMTeacher

Input: Neural Module Teacher $\mathbb{G}_i$, Instance (q, c), Variable Candidates, Beam Width w, Threshold t
$m =$ number of variables in $\mathbb{G}_i$
Initialize PREVSTATES.
for $j = 1$ **to** m **do**
  CURRENTSTATES $\leftarrow \varnothing$
  for STATE in PREVSTATES **do**
    $V \leftarrow$ next unfilled variable
    for CANDIDATE in (CANDIDATES for V) **do**
      Fill V in STATE
      $z \leftarrow$ confidence score of evaluating STATE with $\mathbb{G}_i$
      if $z > t$ **then**
        CURRSTATES.append(STATE)
  Sort (descending) CURRSTATES by z
  PREVSTATES $\leftarrow$ top w states in CURRSTATES
return CURRSTATES

C. Beam Search Algorithm for Neural Module Teacher

In Sec. 3.3 we mentioned the usage of beam search algorithm to search for the best combination of variable assignments. We provide the details in the Algorithm 2.

D. Reproducibility

Computing Infrastructure. Based on GPU availability, we train our models on either Quadro RT 6000, GeForce RTX 2080 Ti or GeForce GTX 1080 Ti. All of our models are trained on single GPU. NMTeacher-Joint requires optimizing both NMTeacher modules and MRC models, so we use Quadro RT 6000 for related experiments.

Number of Parameters. The two trainable modules (FILL and FIND) adopt BERT-base as backbone, using 110 million parameters for each. We use several downstream MRC models in our experiments, and BERT-large is the biggest among all (340 million). To sum, NMTeacher-Joint uses 560 million parameters at most.

Hyper-parameters. We use Adam with linear warmup as our optimizer and we tuned learning rate in the range of $\{1e-5, 2e-5, 3e-5, 4e-5, 5e-5\}$. We set the warmup steps to be either 100 or 500. We tuned the loss balancing co-efficient β (in $\mathcal{L}_{ap}$ and $\mathcal{L}_{au}$, see Sec. 4.2) in the range of $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. We adopt a greedy tuning strategy: first select the best learning rate and fix it; then select the best co-efficient β. We select parameters based on F1 score on dev set.

We set the rotation interval r (see Sec. 4.2) to be 8. We use batch size of 12 for BERT-l; 16 for BERT-b; 16 for ALBERT-b. Gradient accumulation is adopted to achieve such batch size with GPU memory constraint.

Datasets. We download both datasets we use from official websites. SQuAD: https://rajpurkar.github.io/SQuAD-explorer/; Natural Questions: https://ai.google.com/research/NaturalQuestions/download. Note that we customized the settings of NQ dataset as we limit our scope to MRC task. We aim to analysis the capability of NMTeacher in different scenarios, and thus we choose not to use the official test set due to submission constraints (*e.g.*, one attempt per week). We create our own dev and test set (see Sec. 5.1).

Development Set Performance. Table 4 and 6 in the main paper lists test set performance, while their corresponding development set performance can be found in Table 11 and 12.

E. Explanation Collection

Our interface for collecting semi-structured explanations with Amazon Mechanical Turk is shown in Figure 7. Annotators are required to first read a short paragraph of high-level instructions and then read five provided examples. After that, they are required to write an explanation for a provided answered (q, c, a) triple in one single text input box, using suggested expressions in a provided table. Finally, annotators are required to double-check their explanation before they submit. The reward for each accepted explanation is $\$0.5$.

We automatically rejected responses not following instructions (e.g., not mentioning any variables, quoted words do not appear in context). Statistics of the collected explanations on SQuAD and NQ datasets are previously shown in Table 5. We constructed and modified our parser simultaneously with the explanation collection process. The accuracy of semantic parsing is 91.93% by manual inspection on 35 parsed explanations (161 sentences).

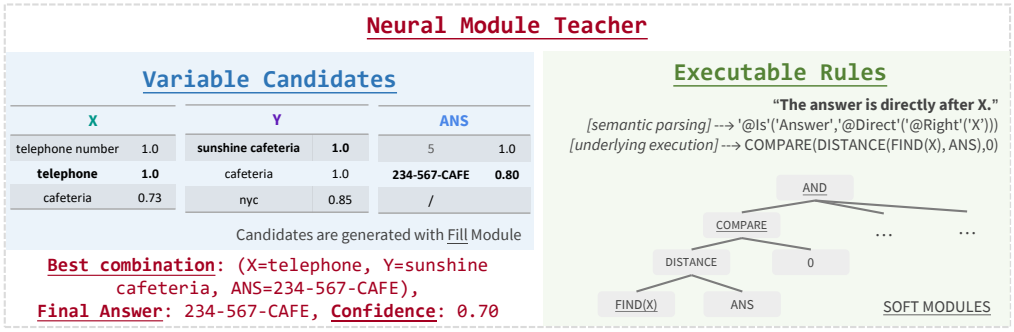

Figure 6. **Example for Beam Search and Extracting an Answer.** Candidates are proposed by Fill module. The best combination is selected by ranking and conducting beam search on possible combinations. Ranking is done by softened execution of rules.

#Explanations ($\lvert\mathcal{S}_a\rvert$, $\lvert\mathcal{S}_p\rvert$)	13 (131, 314)		26 (424, 1048)		52 (766, 2329)	
	EM	F1	EM	F1	EM	F1
BiDAF ($\mathcal{S}_a$)	3.68 ± 0.82	7.40 ± 0.61	4.68 ± 0.57	9.39 ± 0.22	8.31 ± 0.55	13.99 ± 1.01
+ NMTeacher-DA ($\mathcal{S}_p$)	4.89 ± 0.18	8.31 ± 0.12	6.24 ± 0.07	11.29 ± 0.20	13.58 ± 1.51	21.80 ± 2.15
BERT-b ($\mathcal{S}_a$)	11.70 ± 0.88	19.11 ± 1.28	22.32 ± 0.24	33.11 ± 0.47	32.22 ± 1.81	42.68 ± 2.58
+ NMTeacher-DA ($\mathcal{S}_p$)	15.68 ± 1.10	25.43 ± 0.98	24.88 ± 3.01	35.65 ± 4.63	35.67 ± 3.23	46.86 ± 3.41
BERT-l ($\mathcal{S}_a$)	15.51 ± 1.61	23.65 ± 2.69	29.50 ± 2.00	42.05 ± 2.23	39.03 ± 0.63	51.90 ± 0.52
+ NMTeacher-DA ($\mathcal{S}_p$)	18.67 ± 1.94	30.87 ± 2.84	32.76 ± 2.38	46.52 ± 3.22	43.87 ± 2.36	56.60 ± 2.41
+ Self Training ($\mathcal{S}_u$)	15.59 ± 1.48	23.19 ± 1.78	35.48 ± 7.93	45.07 ± 6.04	46.14 ± 3.30	57.83 ± 3.81
+ Mean Teacher ($\mathcal{S}_u$)	13.28 ± 2.48	21.54 ± 3.15	35.27 ± 4.87	48.35 ± 4.32	45.75 ± 7.14	58.82 ± 5.65
+ Pseudo Labeling, PL ($\mathcal{S}_u$)	15.96 ± 2.45	23.51 ± 3.66	41.36 ± 5.59	53.71 ± 7.26	52.95 ± 2.26	65.10 ± 1.14
+ NMTeacher-Joint ($\mathcal{S}_p + \mathcal{S}_u$)	37.06 ± 13.64	46.83 ± 17.34	**61.27 ± 1.93**	73.71 ± 2.81	62.22 ± 0.46	74.22 ± 1.24
ALBERT-b ($\mathcal{S}_a$)	32.92 ± 1.59	45.62 ± 1.27	43.65 ± 1.63	57.12 ± 2.82	48.81 ± 1.73	62.06 ± 0.17
+ NMTeacher-DA ($\mathcal{S}_p$)	37.66 ± 2.36	50.25 ± 1.99	44.97 ± 1.20	58.60 ± 1.02	51.35 ± 2.07	64.27 ± 0.75
+ Self Training ($\mathcal{S}_u$)	37.67 ± 4.36	48.32 ± 4.74	49.88 ± 3.06	61.81 ± 3.54	52.08 ± 2.45	65.34 ± 2.87
+ Mean Teacher ($\mathcal{S}_u$)	33.16 ± 2.95	45.42 ± 2.01	43.42 ± 2.58	58.14 ± 1.74	56.86 ± 1.75	70.67 ± 1.52
+ Pseudo Labeling, PL ($\mathcal{S}_u$)	31.02 ± 3.32	43.88 ± 4.76	42.63 ± 2.56	55.62 ± 2.72	48.28 ± 1.63	60.45 ± 2.45
+ NMTeacher-Joint ($\mathcal{S}_p + \mathcal{S}_u$)	**42.40 ± 7.47**	**56.60 ± 6.57**	60.21 ± 3.05	**74.44 ± 2.64**	**62.48 ± 1.23**	**75.76 ± 0.77**

Table 11. **Performance on the development set on SQuAD dataset using different numbers of explanations.**

#Explanations ($\lvert\mathcal{S}_a\rvert$, $\lvert\mathcal{S}_p\rvert$)	18 (98, 539)		36 (107, 647)		54 (273, 1047)	
	EM	F1	EM	F1	EM	F1
BERT-l ($\mathcal{S}_a$)	12.33 ± 2.28	22.08 ± 2.55	15.18 ± 0.35	24.89 ± 1.97	14.62 ± 0.77	24.46 ± 1.02
+ NMTeacher-DA ($\mathcal{S}_p$)	17.12 ± 1.04	28.20 ± 0.90	19.60 ± 1.45	31.05 ± 1.70	20.10 ± 1.13	31.48 ± 1.49
+ Self Training ($\mathcal{S}_u$)	15.76 ± 2.07	25.41 ± 0.46	18.61 ± 1.36	27.77 ± 0.31	18.02 ± 1.04	26.18 ± 0.54
+ Mean Teacher ($\mathcal{S}_u$)	15.68 ± 0.74	25.92 ± 0.59	17.41 ± 0.76	27.97 ± 1.11	18.64 ± 1.55	27.88 ± 1.69
+ Pseudo Labeling, PL ($\mathcal{S}_u$)	18.02 ± 2.07	25.64 ± 0.92	20.95 ± 2.52	28.55 ± 2.63	17.17 ± 0.42	26.12 ± 0.48
+ NMTeacher-Joint ($\mathcal{S}_p + \mathcal{S}_u$)	16.69 ± 0.79	28.48 ± 1.16	21.62 ± 1.82	32.51 ± 2.06	22.90 ± 2.24	34.02 ± 2.20
ALBERT-b ($\mathcal{S}_a$)	20.02 ± 2.05	28.80 ± 2.21	22.90 ± 1.74	32.19 ± 2.22	21.65 ± 0.83	32.23 ± 1.20
+ NMTeacher-DA ($\mathcal{S}_p$)	21.27 ± 1.19	30.87 ± 1.99	25.80 ± 2.48	35.92 ± 2.78	23.22 ± 2.73	33.94 ± 2.98
+ Self Training ($\mathcal{S}_u$)	19.68 ± 1.66	28.67 ± 2.09	23.64 ± 2.70	32.73 ± 1.79	23.64 ± 2.36	32.78 ± 2.58
+ Mean Teacher ($\mathcal{S}_u$)	19.44 ± 0.12	28.84 ± 1.04	24.79 ± 2.92	33.96 ± 3.15	29.23 ± 3.63	38.84 ± 3.27
+ Pseudo Labeling, PL ($\mathcal{S}_u$)	19.04 ± 1.29	27.35 ± 2.11	22.98 ± 2.47	31.48 ± 1.29	20.34 ± 0.92	31.07 ± 2.59
+ NMTeacher-Joint ($\mathcal{S}_p + \mathcal{S}_u$)	**24.44 ± 4.08**	**35.09 ± 5.30**	**31.12 ± 2.38**	**41.74 ± 3.56**	**29.13 ± 3.77**	**40.22 ± 3.98**

Table 12. **Performance on development set on Natural Questions dataset using different numbers of explanations.**

Instructions:
Please read carefully to get accepted!
(1) You're **not** required to answer the question. The answer is already provided and marked in red. **Read examples below carefully to learn about what we want!**
(2) Identify **important short phrases** that appear **both** **in the question and in the context**.
Important: The two appearances of the phrase should be **exactly the same** (trivial differences like plural form or past tense are still acceptable).
Important: Write sentences like *Y is "Switzerland"*. Make sure there is **no typo** in what you quote.
(3) **Explain** how you locate the answer with the phrases you marked; **Only** **use the suggested expressions** in the table in the bottom.

Example 1:
Question: How long has Switzerland traditionally been neutral?
Context: Traditionally, Switzerland avoids alliances that might entail military, political, or direct economic action and has been neutral since the end of its expansion in 1515.
Answer: since the end of its expansion in 1515
Explanation: X is "been neutral". Y is "Switzerland". X and Y appear both in the question and in the context. The answer directly follows X. The answer starts with "since".

[4 Examples Omitted Here]

Your turn to write explanations:
Question: who is the author of brave new world
Context: Brave New World is a dystopian novel by English author Aldous Huxley . Published in 1932 , it propounds that economic chaos and unemployment will cause a radical reaction in the form of an international scientific empire that manufactures its citizens in the laboratory on a eugenic basis , without the need for human intercourse .
Answer: Aldous Huxley

You're **required** to **only** use the expressions in the table below.
☐ This question is complicated; I **cannot** explain it with the expressions in the table below. (in this case please also input "None" in the text box below)

Objective	Expression	Example
0. Mark the Phrase	is, means	X is "been neutral". Y is "Switzerland".
1. Relative Position	before, after, between, left, right, follow, precede, sandwich	"may be" is between X and Y. "near" is before X. X and Y sandwich "is".
2. Distance	within, less than, directly	The answer is within 2 words left of X. The answer directly precedes Z.
3. Contain	start with, end with, contain	The answer starts with "by". The question contains "who".
4. Phrase (NER) Type	person, time, location, organization	X should be a person. The answer should be a time.
5. Phrase (chunk) Type	noun, verb, adjective, adverb, prepositional phrase	The answer is a noun. The answer is a prepositional phrase.

Your explanation for the question answering example above: (i.e. How to locate the answer with XYZs?)

Before you submit, double check the following, or you may get rejected.
(1) XYZ are phrases that appear both in the question and the context. There is **no typo** when you quote these phrases.
(2) You explain how to locate the answer with XYZ by **only** **using expressions in the table**.
(3) What you describe sticks to the question answering example on this page.
Thank you!

Submit

Figure 7. **Crowd-sourcing Interface on Amazon Mechanical Turk.** The interface has four parts: (1) High-level instruction; (2) 5 examples; (3) QA instance requiring explanation and an input box; (4) Final check instructions.