LEARNING FROM EXPLANATIONS WITH NEURAL MODULE EXECUTION TREE

Anonymous authors
Paper under double-blind review

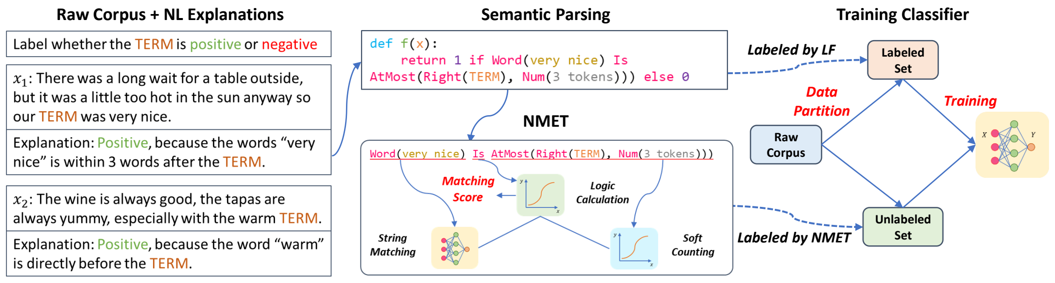
ABSTRACT

While deep neural networks have achieved impressive performance on a range of NLP tasks, these data-hungry models heavily rely on labeled data. To make the most of each example, previous work has introduced natural language (NL) explanations to serve as supplements to mere labels. Such NL explanations can provide sufficient domain knowledge for generating more labeled data over new instances, while the annotation time only doubles. However, directly applying the NL explanations for augmenting model learning encounters two challenges. First, NL explanations are unstructured and inherently compositional, which asks for modularized model to represent their semantics. Second, NL explanations often have large numbers of linguistic variants, resulting in low recall and limited generalization ability when applied to unlabeled data. In this paper, we propose a novel Neural Modular Execution Tree (NMET) framework for augmenting sequence classification with NL explanations. After transforming NL explanations into executable logical forms with a semantic parser, NMET employs a neural module network architecture to generalize different type of actions (specified by the logical forms) for labeling data instances, and accumulates the results with soft logic, which substantially increases the coverage of each NL explanation. Experiments on two NLP tasks, relation extraction and sentiment analysis, demonstrate its superiority over baseline methods by leveraging NL explanation. Its extension to multi-hop question answering achieves performance gain with light annotation effort. Also, NMET achieves much better performance compared to traditional label-only supervised models in the same annotation time.

1 INTRODUCTION

Deep neural networks have achieved state-of-the-art performance on a wide range of natural language processing tasks. However, they usually require massive labeled data, which restricts their applications in scenarios where data annotation is expensive. The traditional way of providing supervision is human-generated labels. For example, in sentiment analysis, given a sentence *"Quality ingredients preparation all around, and a very fair price for NYC"*, an annotator should label it as *"Positive"*. However, the label itself does not provide information about how the decision is made. A more informative method is to allow annotators to explain their decisions in natural language so that the annotation can generalize to other examples. In the above example, an explanation can be *"Positive, because the word price is directly preceded by fair"*, which can generalize to instances like *"Delicious food with a fair price"*. Natural language (NL) explanations have shown effectiveness in providing additional supervision, especially in low-resource settings (Srivastava et al., 2017; Hancock et al., 2018). Also, they can be easily collected from human annotators without significantly increasing their annotation efforts.

However, exploiting NL explanations as supervision is challenging due to the complex nature of human languages. First of all, textual data are not well-structured, and thus we have to parse explanations into logical forms for machine to better utilize them. Also, linguistic variants are ubiquitous, which makes it difficult to generalize an NL explanation for matching sentences that are semantically equivalent but having different word usage. When we perform exact matching with the previous example explanation, it can fail to annotate sentences with "reasonable price" or "good deal".



Label whether the TERM is positive or negative

x_1: There was a long wait for a table outside, but it was a little too hot in the sun anyway so our TERM was very nice.

Explanation: Positive, because the words "very nice" is within 3 words after the TERM.

x_2: The wine is always good, the tapas are always yummy, especially with the warm TERM.

Explanation: Positive, because the word "warm" is directly before the TERM.

```
def f(x):
    return 1 if Word(very nice) Is
AtMost(Right(TERM), Num(3 tokens))) else 0
```

```
Word(very nice) Is AtMost(Right(TERM), Num(3 tokens)))
```

Figure 1: Overview of the NMET Framework. Natural language explanations are firstly parsed into logical forms. Then we partition the raw corpus $\mathcal{S}$ into labeled dataset $\mathcal{S}_a$ and unlabeled dataset $\mathcal{S}_u = \mathcal{S} - \{\mathbf{x}_i^a\}_{i=1}^{N_a}$. We use *matching modules* to provide supervision on $\mathcal{S}_u$. Finally, supervision from both $\mathcal{S}_a$ and $\mathcal{S}_u$ is fed into a classifier.

Attempts have been made to train classifiers with NL explanations. Srivastava et al. (2017) use NL explanations as additional features of data. They map explanations to logical forms with a semantic parser and use them to generate binary features for all instances. Hancock et al. (2018) employ a rule-based semantic parser to get logical forms (i.e. "labeling function") from NL explanations that generate noisy labeled datasets which are used for training models. While both methods claim huge performance improvements, they neglect the importance of linguistic variants, thus resulting in a very low recall. Also, their methods of evaluating explanations on new instances are oversimplified (e.g. comparison/logic operators), making their methods over-confident. In the above example, a sentence like *"Decent sushi at a fair enough price"* will be rejected because of the *"directly preceded"* requirement.

Therefore, we believe that the generalization ability of NL explanations is under-explored. We emphasize that a good data annotation method should 1) be able to generalize annotations to semantically similar instances (beyond stemming, part-of-speech, etc.) and 2) model the uncertainty in annotations. Towards these goals, we propose `Neural Modular Execution Tree` (NMET) framework for learning neural models with explanations, as illustrated in Figure 1. Given a raw corpus and a set of NL explanations, we first parse the NL explanations to machine-actionable logic forms by a combinatory categorial grammar (CCG) based semantic parser. Different from previous work, we "soften" the annotation process by generalizing the predicates using neural module network and changing the labeling process from exact matching to fuzzy matching. We introduce four types of matching modules, namely *String Matching Module*, *Soft Counting Module*, *Logical Calculation Module*, and *Deterministic Function Module*, which serve as matching semantically similar words, scoring different positions, accumulating matching scores from other modules, and dealing with deterministic predicates, respectively. We calculate the matching scores and find for each instance the most similar logical form. Thus, all instances in the raw corpus can be assigned a label and used to train the neural model. We conduct extensive experiments on two representative tasks, relation extraction and sentiment analysis. Experimental results demonstrate the superiority of NMET over various baseline methods. Also, we adapted NMET for multi-hop question answering task, in which it achieves performance improvement with only 21 explanations and 5 rules.

2 Learning to Augment Sequence Models with NL Explanations

We consider the task of training classifiers with natural language explanations for text classification (e.g., relation extraction, sentiment analysis) in a low-resource setting. Specifically, given a raw corpus $\mathcal{S} = \{\mathbf{x}_i\}_{i=1}^{N} \subseteq \mathcal{X}$ and a predefined label set $\mathcal{Y}$, our goal is to learn a classifier $f_c : \mathcal{X} \to \mathcal{Y}$. We ask human annotators to view a subset $\mathcal{S}'$ of the corpus $\mathcal{S}$ and provide for each instance $\mathbf{x} \in \mathcal{S}'$ an explanation $\mathbf{e}$ and a label y, which explains why the instance should receive that label. Note that $|\mathcal{S}'| \ll |\mathcal{S}|$, which requires our framework to learn with very limited human supervision.

Approach Overview. We develop a multi-stage learning framework to leverage NL explanations in weakly-supervised setting. An overview of our framework is depicted in Fig. 1. Our NMET framework consists of three stages, namely explanation parsing stage, dataset partition stage, and model learning stage. Human explanations are first converted to machine-actionable logical forms by a semantic parser. Then those extracted logical forms are used to label the raw corpus by performing exact matching. After that we have a labeled dataset and an unlabeled dataset. For unlabeled data, we propose Neural Module Execution Tree (NMET) to relax the constraints and generalize the keywords in the logical forms. Therefore, they can be used to evaluate on unlabeled instances, and assign them with pseudo labels accompanied by confidence scores. The task-specific classifier is then jointly optimized with NMET over labeled data and pseudo-labeled data.

Explanation Parsing. To leverage the unstructured human explanations $\mathcal{E} = \{\mathbf{e}_j\}_{j=1}^{|\mathcal{S}'|}$, we turn them into logical forms (i.e., labeling functions) (Ratner et al., 2016), which can be denoted as $\mathcal{F} = \{f_j : \mathcal{X} \to \{0,1\}\}_{j=1}^{|\mathcal{S}'|}$, where 1 indicates the the logical form matches the input sequence and 0 otherwise. To access the labels, we introduce a function $h : \mathcal{F} \to \mathcal{Y}$ that maps each logical form f_j to the label y_j of its explanation $\mathbf{e}_j$. Examples are given in Fig. 1. We use Combinatory Categorial Grammar (CCG) based semantic parsing (Zettlemoyer & Collins, 2012; Artzi et al., 2015), an approach that couples syntax with semantics, to convert each NL explanation $\mathbf{e}_j$ to a logical form f_j.

Following Srivastava et al. (2017), we first compile a domain lexicon that maps each word to its syntax and logical predicate. Frequently-used predicates are listed in the Appendix. For each explanation, the parser is able to generate many possible logical forms based on CCG grammar. To identify the correct one from these logical forms, we use a feature vector $\boldsymbol{\phi}(f) \in \mathcal{R}^d$ with each element counting the number of applications of a particular CCG combinator (similar to Zettlemoyer & Collins (2007)). Specifically, given an explanation $\mathbf{e}_i$, the semantic parser parameterized by $\boldsymbol{\theta} \in \mathcal{R}^d$ outputs a probability distribution over all possible logical forms $\mathcal{Z}_{\mathbf{e}_i}$. The probability of a feasible logical form can be calculated as:

$$P_\theta(f|\mathbf{e}_i) = \frac{\exp \boldsymbol{\theta}^T \boldsymbol{\phi}(f)}{\sum_{f':f' \in \mathcal{Z}_{\mathbf{e}_i}} \exp \boldsymbol{\theta}^T \boldsymbol{\phi}(f')}.$$

To learn θ, we maximize the probability of $\mathbf{y}_i$ given $\mathbf{e}_i$ calculated by marginalizing over all logical forms that match $\mathbf{x}_i$ (similar to Liang et al. (2013)). Formally, the objective function is defined as:

$$L_{parser} = \sum_{i=1}^{|\mathcal{S}'|} \log \big(\sum_{f:f(\mathbf{x}_i)=1 \,\wedge\, h(f)=y_i} P_\theta(f|\mathbf{e}_i) \big).$$

When the optimal θ^* is derived using gradient-based method, the parsing result for $\mathbf{e}_i$ is defined as $f_i = \arg\max_f P_{\theta^*}(f|\mathbf{e}_i)$.

Dataset Partition. After we parse explanations $\{\mathbf{e}_i\}_{i=1}^{|\mathcal{S}'|}$ into $\mathcal{F} = \{f_i\}_{i=1}^{|\mathcal{S}'|}$ where each f_i corresponds to $\mathbf{e}_i$, we use $\mathcal{F}$ to find exact matches in $\mathcal{S}$ and pair them with the corresponding labels. We denote the number of instances getting labeled by exact matching as N_a. As a result, $\mathcal{S}$ is partitioned into a labeled dataset $\mathcal{S}_a = \{(\mathbf{x}_i^a, y_i^a)\}_{i=1}^{N_a}$, and an unlabeled dataset $\mathcal{S}_u = \mathcal{S} - \{\mathbf{x}_i^a\}_{i=1}^{N_a} = \{\mathbf{x}_j^u\}_{j=1}^{N_u}$ where $N_u = |\mathcal{S}| - N_a$.

Model Learning with Neural Module Execution Tree. So far we can already make use of exact-matched $\mathcal{S}_a$ to train a classifier; however, informative instances in $\mathcal{S}_u$ are left untouched. We propose Neural Module Execution Tree (NMET), which relaxes constraints in each logic form f_j and substantially improve the rule coverage in $\mathcal{S}_u$. Classifiers will benefit from these soft-matched and pseudo-labeled instances. Trainable parameters in NMET are also jointly optimized with the classifier. Details of NMET and joint training will be introduced in next section.

3 Neural Module Execution Tree

Given a logical form f and a sentence $\mathbf{x}$, Neural Module Execution Tree (NMET) will output a matching score $u_s \in [0,1]$, which indicates how likely the sentence $\mathbf{x}$ satisfies the logical form f and thus should be given the corresponding label $h(f)$. Specifically, NMET comprises of four modules to deal with four categories of predicates, namely *String Matching Module*, *Soft Counting Module*, *Deterministic Function Module*, and *Logical Calculation Module*. Any complex logical form can be disassembled into clauses containing these four categories of predicates. The four modules are then used to evaluate each clause and then the whole logical form in a softened way.

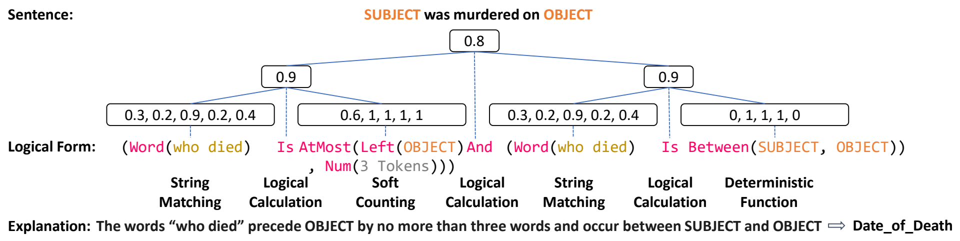

Sentence:
SUBJECT was murdered on OBJECT
0.8
0.9
0.9
0.3, 0.2, 0.9, 0.2, 0.4
0.6, 1, 1, 1, 1
0.3, 0.2, 0.9, 0.2, 0.4
0, 1, 1, 1, 0
Logical Form:
(Word(who died) Is AtMost(Left(OBJECT) And (Word(who died) Is Between(SUBJECT, OBJECT))
, Num(3 Tokens)))
String Matching
Logical Calculation
Soft Counting
Logical Calculation
String Matching
Logical Calculation
Deterministic Function
Explanation: The words "who died" precede OBJECT by no more than three words and occur between SUBJECT and OBJECT ⇒ Date_of_Death

Figure 2: Neural Module Execution Tree (NMET) softly executes the logical form on the sentence.

Fig. 2 shows how NMET builds the execution tree from an NL explanation and how it evaluates an unlabeled sentence. We show in the figure the corresponding module for each predicate in the logical form. We introduce the four modules respectively as follows.

3.1 Modules in NMET

String Matching Module. Given a keyword query $\mathbf{q}$ derived from an explanation and an input sequence $\mathbf{x} = [w_1, w_2, ..., w_n]$, the string matching module $f_s(\mathbf{x}, \mathbf{q})$ returns a sequence of scores $[s_1, s_2, ..., s_n]$ indicating the similarity between each token w_i and the query $\mathbf{q}$. Previous work implements this operation by exact keyword searching, while we augment the module with neural networks to enable capturing semantically similar words. Inspired by Li et al. (2018), for token w_i, we first generate N_c contexts by sliding windows of various lengths. For example, if the maximum window size is 2, the contexts $\mathbf{c}_{i0}, \mathbf{c}_{i1}, \mathbf{c}_{i2}$ of token w_i are $[w_i]$, $[w_{i-1}; w_i]$ and $[w_i; w_{i+1}]$ respectively. Then we encode each context $\mathbf{c}_{ij}$ to a vector $\mathbf{z}_{\mathbf{c}_{ij}}$ by feeding pre-trained word embeddings into a bi-directional LSTM encoder (Hochreiter & Schmidhuber, 1997) followed by an attention layer (Bahdanau et al., 2014). For keyword query $\mathbf{q}$, we directly encode it into vector $\mathbf{z}_q$ by bi-LSTM and attention. Finally, scores of sentence $\mathbf{x}$ and query $\mathbf{q}$ are calculated by aggregating similarity scores from different sliding windows:

$$\boldsymbol{M}_{ij}(\mathbf{x}, \mathbf{q}) = \cos(\mathbf{z}_{\mathbf{c}_{ij}} \boldsymbol{D}, \mathbf{z}_{\mathbf{q}} \boldsymbol{D}), \quad f_s(\mathbf{x}, \mathbf{q}) = \boldsymbol{M}(\mathbf{x}, \mathbf{q})\mathbf{v},$$

where $\boldsymbol{D}$ is a trainable diagonal matrix, $\mathbf{v} \in \mathcal{R}^{N_c}$ is the trainable weight of each sliding window.

Parameters in the string matching module need to be learned with data in the form of *(sentence, keyword, label)*. To build a training set for learning string matching, we randomly select spans of consecutive words as keyword queries in the training data. Each query is paired with either the sentence it comes from or a different sentence. The synthesized dataset is denoted as $\{\mathbf{x}_i, \mathbf{q}_i, \mathbf{k}_i\}_{i=1}^{N_{syn}}$, where $\mathbf{k}_{ij}$ will take the value of 1 if $\mathbf{q}$ is extracted from $\mathbf{x}_{ij}$ and 0 otherwise. The loss function is defined as the binary cross entropy loss, as follows.

$$L_{find} = -\frac{1}{N_{syn}} \sum_{i=1}^{N_{syn}} \frac{1}{|\mathbf{k}_i|} \cdot (\mathbf{k}_i \log f_s(\mathbf{x}_i, \mathbf{q}_i) + (1 - \mathbf{k}_i) \log(1 - f_s(\mathbf{x}_i, \mathbf{q}_i))).$$

While pretraining with L_{find} enables matching similar words, this unsupervised distributional method is poor at learning their semantic meanings. For example, the word "good" will have very high similarity to "bad" because they often coexist with the same set of context words. To solve this problem, we borrow the idea of word retrofitting (Faruqui et al., 2014) and adopt a contrastive loss (Neculoiu et al., 2016) to incorporate semantic knowledge in training. We use the keyword queries in labeling functions as supervision. Intuitively, the semantic meaning of two queries should be similar if they appear in the same class of labeling functions and dissimilar otherwise. More specifically, for a query $\mathbf{q}$, we denote queries in the same class of labeling functions as $\mathcal{Q}_+$ and queries in different classes of labeling functions as $\mathcal{Q}_-$. The similarity loss is defined as:

$$L_{sim} = \max_{\mathbf{q}_1 \in \mathcal{Q}_+} \{(\tau - \cos(\mathbf{z}_{\mathbf{q}} \boldsymbol{D}, \mathbf{z}_{\mathbf{q}_1} \boldsymbol{D}))_+^2\} + \max_{\mathbf{q}_2 \in \mathcal{Q}_-} \{\cos(\mathbf{z}_{\mathbf{q}} \boldsymbol{D}, \mathbf{z}_{\mathbf{q}_2} \boldsymbol{D})_+^2\}.$$

The overall objective function for string matching module is:

$$L_{string} = L_{find} + \gamma \cdot L_{sim}, \tag{1}$$

where γ is a hyper-parameter. We pretrain the string matching module for better initialization.

Algorithm 1: Learning on Unlabeled Data with NMET

```
Input: Labeled data $\mathcal{S}_a = \{(\mathbf{x}_i^a, y_i^a)\}_{i=1}^{N_a}$, unlabeled data $\mathcal{S}_u = \{\mathbf{x}_j^u\}_{j=1}^{N_u}$, and logical forms $\mathcal{F} = \{f_k\}_{k=1}^{|\mathcal{S}'|}$.
Output: A classifier $f_c : \mathcal{X} \to \mathcal{Y}$.
Pretrain String Matching Module in NMET w.r.t. $L_{string}$ using Eq. 1.
while not converge do
    Sample a labeled batch $\mathcal{B}_a = \{(\mathbf{x}_i^a, y_i^a)\}_{i=1}^{n}$ from $S_a$, and an unlabeled batch $\mathcal{B}_u = \{\mathbf{x}_j^u\}_{j=1}^{m}$ from $\mathcal{S}_u$.
    foreach $\mathbf{x}_j^u \in \mathcal{B}_u$ do
        Calculate a pseudo label $y_j^u$ for $\mathbf{x}_j^u$ with confidence $u_j$ using NMET and $\mathcal{F}$.
    Normalize matching scores $\{u_j\}_{j=1}^m$ to get $\{\omega_j\}_{j=1}^m$ based on Eq. 3.
    Calculate $L_a$ using Eq. 2, $L_u$ using Eq. 4, $L$ using Eq. 5.
    Update $f_c$ and String Matching Module in NMET w.r.t. $L_{total}$.
```

Soft Counting Module. The soft counting module aims to relax the counting (distance) constraints defined by NL explanations. For a counting constraint *precede object by no more than three words*, the soft counting module outputs a matching score indicating to what extent an anchor word (TERM, SUBJECT and OBJECT) satisfies the constraint. The score is set to 1 if the position of the anchor word strictly satisfies the constraint, and will decrease if the constraint is broken. For simplicity, we allow an additional range in which the score is set to $\mu \in (0, 1)$, which is a hyper-parameter controlling the constraints.

Deterministic Function Module. The deterministic function module deals with the deterministic predicates like "Between", "Left" and "Right". It outputs a mask sequence where the span satisfying the constraint is marked as 1. Other positions are marked as 0.

Logical Calculation Module. The logical calculation module acts as a score aggregator. It can aggregate scores given by: (1) a string matching module and a soft counting module/deterministic function module (triggered by predicates such as "Is" and "Occur") (2) two clauses that have been evaluated with a score respectively (triggered by predicates such as "And" and "Or").

In the first case, the logical calculation module will calculate the element-wise products of the score sequence provided by the string matching module and the mask sequence provided by the soft counting module/deterministic function module. It then uses max pooling to calculate the matching score of the current clause. In the second case, the logical calculation module will aggregate the scores of at least one clause based on the logic operation. The rules are defined as follows.

$$p_1 \wedge p_2 = \max(p_1 + p_2 - 1, 0), \quad p_1 \vee p_2 = \min(p_1 + p_2, 1), \quad \neg p = 1 - p,$$

where p is the score of the input clause.

3.2 Augmenting Model Learning with NMET

As described in Algo. 1, in each iteration, we sample two batches $\mathcal{B}_a$ and $\mathcal{B}_u$ from $\mathcal{S}_a$ and $\mathcal{S}_u$. We conduct supervised learning on $\mathcal{B}_a$. The labeled loss function is calculated as:

$$L_a = -\frac{1}{N_a} \sum_{(\mathbf{x}_i^a, y_i^a) \in \mathcal{B}_a} \log p(y_i^a | \mathbf{x}_i^a). \tag{2}$$

To leverage $\mathcal{B}_u$, which is also informative, for each instance $\mathbf{x}_j^u \in \mathcal{B}_u$, we can use our *matching modules* to compute its matching score with every logical form. The most probable logical form that matched with $\mathbf{x}_j^u$ is denoted as $\mathbf{y}_j^u$ [1], along with the matching score u_j. To ensure the scale of the unlabeled loss is comparable to labeled loss, we normalize the matching scores among pseudo-labeled instances in $\mathcal{B}_u$ as:

$$\omega_j = \frac{\exp(\theta_t u_j)}{\sum_{k=1}^{|\mathcal{B}_u|} \exp(\theta_t u_k)}, \tag{3}$$

[1] *None* label (e.g. *No_Relation* for relation extraction and *neutral* for sentiment analysis) usually lacks explanations and logical forms. If the entropy of matching score distribution over labels is higher than a threshold, a *None* label will be given.

Dataset	exps	categs	avg ops	logic/%	assertion/%	position/%	counting/%	acc/%
TACRED	170	13	8.2	25.8	21.3	21.4	12.4	95.3
SemEval	203	9	4.2	32.7	15.9	26.3	5.5	84.2
Laptop	40	8	3.9	0.0	23.8	23.8	17.5	87.2
Restaurant	45	9	9.6	2.8	25.4	26.1	16.2	88.2

Table 1: **Statistics for Human-curated Explanations and Evaluation of Semantic Parsing.** We report the number of NL explanations (*exps*), categories of predicates (*categs*) and operator compositions per explanation (*avg ops*) respectively. We also report the proportions of different types of predicates, where *logic* denotes logical operators (*And*, *Or*), *assertion* denotes assertion predicates (*Occur*, *Contains*), *position* denotes position predicates (*Right*, *Between*) and *counting* denotes counting predicates (*MoreThan, AtMost*). We summarize the accuracy (acc) of semantic parsing based on human evaluation.

where k is the index of the instance and hyperparameter θ_t (temperature) controls the shape of normalized scores' distribution. Based on that, the unlabeled loss is calculated as:

$$L_u = -\sum_{(\mathbf{x}_j^u \in \mathcal{B}_u)} \omega_j \log p(y_j^u|\mathbf{x}_j^u). \tag{4}$$

Note that the string matching module is also trainable and plays a vital role in NMET. We jointly learn it with the classifier by optimizing:

$$L_{total} = L_a + \alpha \cdot L_u + \beta \cdot L_{string}, \tag{5}$$

where α and β are hyper-parameters.

4 Experiments

Tasks and Datasets. We conduct experiments on two tasks: relation extraction and aspect-term-level sentiment analysis. Relation extraction (RE) aims to identify the relation type between two entities in a sentence. For example, given a sentence *Steve Jobs founded Apple Inc*, we want to extract a triple (*Steve Jobs*, *Apple Inc.*, *Founder*). For RE we choose two datasets, TACRED (Zhang et al., 2017) and SemEval (Hendrickx et al., 2009) in our experiments. Aspect-term-level sentiment analysis (SA) aims to decide the sentiment polarity with regard to the given aspect term. For example, given a sentence *Quality ingredients preparation all around, and a very fair price for NYC*, the sentiment polarity of the aspect term *price* is positive, the explanation can be *The word price is directly preceded by fair*. For this task we use two customer reviews datasets, Restaurant and Laptop, which are part of SemEval 2014 Task 4.

Explanation Collection. We use Amazon Mechanical Turk to collect labels and explanations for a randomly sampled set of instances in each dataset. Turkers are prompted with a list of selected predicates (see Appendix) and several examples of NL explanation. Examples of collected explanations are listed in Appendix. Statistics of curated explanations and intrinsic evaluation results of semantic parsing are summarized in Table 1. To ensure a low-resource setting (i.e., $|\mathcal{S}'| \ll |\mathcal{S}|$), in each experiment we only use a random subset of collected explanations.

Compared Methods. As mentioned in Sec. 2, logic forms partitioned unlabeled corpus $\mathcal{S}$ into labeled set $\mathcal{S}_a$ and unlabeled set $\mathcal{S}_u$. Labeled set $\mathcal{S}_a$ can be directly utilized by supervised learning methods. (1) **CBOW-GloVe** uses bag-of-words (Mikolov et al., 2013) on GloVe embeddings (Pennington et al., 2014) to represent an instance, or surface patterns in NL explanation. It then annotates the sentence with the label of it most similar surface pattern (as with cosine similarity). (2) **PCNN** (Zeng et al., 2015) uses piece-wise max-pooling to aggregate CNN-generated features. (3) **LSTM+ATT** (Bahdanau et al., 2014) adds an attention layer onto LSTM to encode an sequence. (4) **PA-LSTM** (Zhang et al., 2017) combines LSTM with an entity-position aware attention to conduct relation extraction. (5) **ATAE-LSTM** (Wang et al., 2016) combines the aspect term information into both embedding layer and attention layer to help the model concentrate on different parts of a sentence.

In semi-supervised baselines, unlabeled data $\mathcal{S}_u$ are also introduced to training. For methods requiring rules as input, we use surface pattern-based rules transferred from explanations. Compared semi-supervised methods include: (1) **Pseudo-Labeling** (Lee, 2013) first trains a classifier on labeled dataset, then generate pseudo labels for unlabeled data using the classifier by selecting the label with maximum predicted probability. (2) **Self-Training** (Rosenberg et al., 2005) proposes to expand the labeled data by selecting a batch of unlabeled data that has the highest confidence and generate pseudo-labels for them. The method stop until the unlabeled data is used up. (3) **Mean-Teacher** (Tarvainen & Valpola, 2017) averages model weights instead of label predictions and assumes similar data points should have similar outputs. (4) **DualRE** (Lin et al., 2019) jointly trains a relation prediction module and a retrieval module.

Learning from explanations is categorized as a third setting. Both methods generate explanation-guided pseudo labels for a downstream classifier. (1) **Data Programming** (Hancock et al., 2018; Ratner et al., 2016) aggregates results of strict labeling functions for each instance and use these pseudo-labels to train a classifier. (2) **NMET** (proposed work) softly applies logic forms to get annotations for unlabeled instances and train a downstream classifier with these pseudo-labeled instances. The downstream classifier is BiLSTM+ATT for relation extraction and ATAE-LSTM for sentiment analysis.

4.1 Results Overview

Table 2 (a) lists F1 scores of all relation extraction models. Full results including precision and recall can be found in Appendix A.4. We observe that our proposed NMET consistently outperform all baseline models in low-resource setting. Also, we found that, (1) Directly applying logic forms to unlabeled data results in poor performance. We notice that this method achieves high precision but low recall, as expected. (2) Compared to its downstream classifier baseline (BiLSTM+ATT with $\mathcal{S}_l$), NMET achieves 4.2% F1 improvement in absolute value on TACRED, and 5.5% on SemEval. This validates that the expansion of rule coverage by NMET is effective and is providing useful information to classifier training. (3) Performance gap further widens when we take annotation efforts into account. The annotation time for $\mathcal{E}$ and $\mathcal{S}_a$ are equivalent; but the performance of BiLSTM+ATT significantly degrades with fewer instances in $\mathcal{S}_a$. (4) Results of semi-supervised methods are unsatisfactory. This may be explained with difference between underlying data distribution of $\mathcal{S}_a$ and $\mathcal{S}_u$.

Table 2 (b) lists the performances of all sentiment analysis models. The observations are similar to those of relation extraction, which strengthen our conclusions and validates the capability of NMET.

	TACRED	SemEval
LF ($\mathcal{E}$)	23.33	33.86
CBOW-GloVe ($\mathcal{R}+\mathcal{S}$)	34.6±0.4	48.8±1.1
PCNN ($\mathcal{S}_a$)	34.8±0.9	41.8±1.2
PA-LSTM ($\mathcal{S}_a$)	41.3±0.8	57.3±1.5
BiLSTM+ATT ($\mathcal{S}_a$)	41.4±1.0	58.0±1.6
BiLSTM+ATT ($\mathcal{S}_l$)	30.4±1.4	54.1±1.0
Self Training ($\mathcal{S}_a+\mathcal{S}_u$)	41.7±1.5	55.2±0.8
Pseudo Labeling ($\mathcal{S}_a+\mathcal{S}_u$)	41.5±1.2	53.5±1.2
Mean Teacher ($\mathcal{S}_a+\mathcal{S}_u$)	40.8±0.9	56.0±1.1
Mean Teacher ($\mathcal{S}_l+\mathcal{S}_{lu}$)	25.9±2.2	52.2±0.7
DualRE ($\mathcal{S}_a+\mathcal{S}_u$)	32.6±0.7	61.7±0.9
Data Programming ($\mathcal{E}+\mathcal{S}$)	30.8±2.4	43.9±2.4
NMET ($\mathcal{E}+\mathcal{S}$)	**45.6±0.4**	**63.5**±**1.0**

(a) Relation Extraction

	Restaurant	Laptop
LF ($\mathcal{E}$)	7.7	13.1
CBOW-GloVe ($\mathcal{R}+\mathcal{S}$)	68.5±2.9	61.5±1.3
PCNN ($\mathcal{S}_a$)	72.6±1.2	60.9±1.1
ATAE-LSTM ($\mathcal{S}_a$)	71.1±0.4	56.2±3.6
ATAE-LSTM ($\mathcal{S}_l$)	71.4±0.5	52.0±1.4
Self Training ($\mathcal{S}_a+\mathcal{S}_u$)	71.2±0.5	57.6±2.1
Pseudo Labeling ($\mathcal{S}_a+\mathcal{S}_u$)	70.9±0.4	58.0±1.9
Mean Teacher ($\mathcal{S}_a+\mathcal{S}_u$)	72.0±1.5	62.1±2.3
Mean Teacher ($\mathcal{S}_l+\mathcal{S}_{lu}$)	74.1±0.4	61.7±3.7
Data Programming ($\mathcal{E}+\mathcal{S}$)	71.2±0.0	61.5±0.1
NMET ($\mathcal{E}+\mathcal{S}$)	**75.8±0.8**	**62.8±1.9**

(b) Sentiment Analysis

Table 2: **Experiment results on Relation Extraction and Sentiment Analysis.** Average and standard deviation of F1 scores (%) over multiple runs are reported (5 runs for RE and 10 runs for SA). Bracket behind each method illustrates corresponding data used in the method. $\mathcal{S}$ denotes training data without labels, $\mathcal{E}$ denotes explanations, $\mathcal{R}$ denotes surface pattern rules transformed from explanations; $\mathcal{S}_a$ denotes labeled data annotated with explanations, $\mathcal{S}_u$ denotes the remaining unlabeled data. $\mathcal{S}_l$ denotes labeled data annotated using same time as creating explanations $\mathcal{E}$, $\mathcal{S}_{lu}$ denotes remaining unlabeled data corresponding to $\mathcal{S}_l$.

4.2 Performance Analysis

Effectiveness of softening logical rules. As shown in Table 3, we conduct ablation studies on TACRED and Restaurant. We remove two modules that support soft logic (by only allowing them to give 0/1 outputs) to see how much does rule softening help in our framework. Both soft counting module and string matching module contribute to the performance of NMET. It can be easily concluded that string matching module plays a vital role. Removing it leads to significant performance drops, which demonstrates the effectiveness of generalizing when applying logical forms. Besides, we examine the impact brought by L_{sim}, L_{find}. Removing them severely hurts the performance, indicating the importance of semantic learning when performing fuzzy matching.

	TACRED	SemEval	Restaurant	Laptop
Full NMET	45.6±0.4	63.5±1.0	75.8±0.8	62.8±1.9
No counting	44.6±0.9	63.2±0.7	75.6±0.8	62.4±1.9
No matching	41.8±1.1	54.6±1.2	71.2±0.4	57.0±2.7
No L_{sim}	42.5±1.0	56.2±2.9	70.7±0.8	59.4±0.7
No L_{find}	43.2±1.3	60.2±0.9	70.0±3.5	58.1±2.8

Table 3: Ablation study on modules of NMET and losses for string matching module. F1 score on the test set is reported. We remove soft counting module (No counting) and string matching module (No matching) by only allowing them to give 0/1 results.

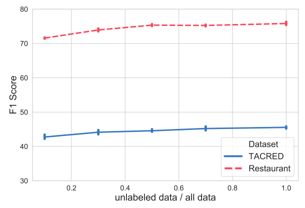


Figure 3: NMET's performance w.r.t. number of unlabeled instances.

Superiority of explanations in data efficiency. In the real world, a more realistic problem is that, with limited human-power, should we just annotate more labels or spend time explaining existing annotations. To answer this question, we carefully examine the data efficiency of explanations over pure labels. For skilled annotators, the time for labeling one instance in TACRED plus giving an explanation is as 2 times as the time for only labeling one instance. For Restaurant, the ratio is 2.3. We then compare the performance between these two labeling strategies within equal annotation time, e.g. 100 explanations v.s. 200 labeled data in TACRED. From Fig. 5 (a) and (b), we can see that our NMET beats purely-supervised and semi-supervised baselines in most cases, even though they are given more labels. We argue that giving explanations surely yields additional cost, but logical form based matching has high accuracy and can generalize well with our NMET. Explanations prove to be an very efficient form for data annotation.

Performance with different number of explanations. From Fig. 5 (c) and (d), one can clearly observe that all approaches benefit from more labeled data. Our NMET outperforms all other baselines by a large margin, which indicates the effectiveness of leveraging knowledge embedded in ML explanations.

Performance with different amount of unlabeled data. To investigate how our NMET's performance is affected by the amount of unlabeled data, we randomly sample 10%, 30%, 50% and 70% of the original unlabeled dataset to do the experiments. As illustrated in Fig. 3, our NMET benefits from larger amount of unlabeled data. We attribute it to high accuracy of logical forms converted from explanations.

Case study on string matching module. String matching module plays a vital rule in NMET. The matching quality greatly influences the accuracy of pseudo labeling. In Fig. 4, we can see that keyword *chief executive of* is perfectly aligned with *executive director of* in the sentence, which demonstrates the effectiveness of string matching module in capturing semantic similarity.

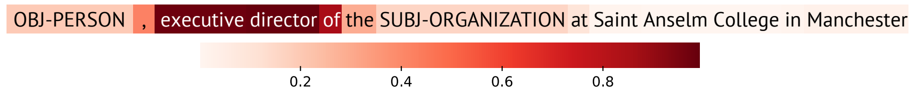
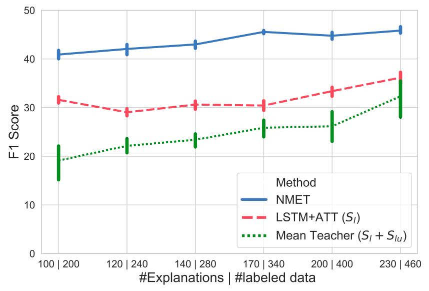
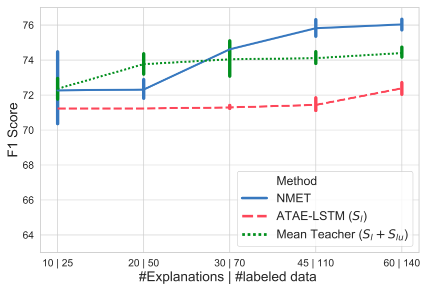


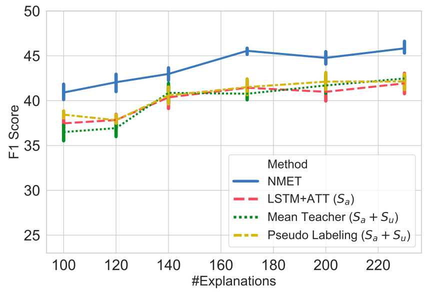
Figure 4: Heatmap for keyword *chief executive of* and sentence *OBJ-PERSON, executive director of the SUBJ-ORGANIZATION at Saint Anselm College in Manchester*. Results show that our string matching module can successfully grasp relevant words.



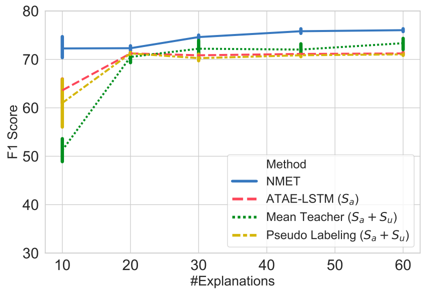
(a) TACRED



(b) Restaurant



(c) TACRED



(d) Restaurant

Figure 5: **Study on Data Efficiency**: (a) and (b), and **Performance with different number of explanations**: (c) and (d). We choose supervised semi-supervised baselines for comparison.

4.3 Additional Experiment on Multi-hop Reasoning

To further test the capability of NMET in downstream tasks, we apply it to WikiHop (Welbl et al., 2018) 'country' task by fusing NMET-matched facts into baseline model NLProlog (Weber et al., 2019). For a brief introduction, WikiHop is a multi-hop question answering (QA) dataset that requires a model to select the correct *entity2* in statement *(entity1, predicate, entity2)* given a candidate pool and several support sentences. NLProlog considers entity-masked support sentence as relation, calculates relation-predicate similarity and entity-entity similarity with mapped Sent2Vec embeddings (Pagliardini et al., 2018), and use these similarity scores for weak unification to solve candidate statements with a Prolog solver.

Fig. 6 shows how the framework in Fig. 1 is adjusted to suit NLProlog. We manually choose 3 predicates (i.e., located_in, capital_of, next_to) and annotate 21 support sentences with natural language explanation. We get 103 strictly-matched facts ($\mathcal{S}_a$) and 1407 NMET-matched facts ($\mathcal{S}_u$) among the 128k unlabeled QA support sentences. Additionally, we manually write 5 rules about these 3 predicates for the Prolog solver, e.g. located_in(X,Z) $\leftarrow$ located_in(X,Y) $\wedge$ located_in(Y,Z).

Results are listed in Table 4. From the result we observe that simply adding the 103 strictly-matched facts is not making notable improvement. However, with the help of NMET, a larger number of structured facts are recognized from support sentences, so that external knowledge from only 21 explanations and 5 rules improve the accuracy by 1 point. This observation validates NMET's capability in low resource setting and highlight its potential when applied to downstream tasks.

5 Related Work

Leveraging natural language for training classifiers. Supervision in the form of natural language has been explored by many works. Srivastava et al. (2017) first demonstrates the effectiveness of NL explanations. They proposed a joint concept learning and semantic parsing method for classification problems. However, the method is very limited in that it is not able to use unlabeled data. To address this issue, Hancock et al. (2018) propose to parse the NL explanations into labeling functions and then use data programming to handle the conflict and enhancement between different labeling

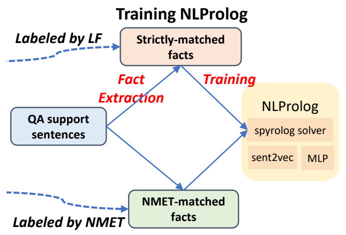


Figure 6: Adjusting NMET Framework (Fig. 1) for NLPROLOG.

	$\|\mathcal{S}_a\|$	$\|\mathcal{S}_u\|$	Accuracy
NLProlog (published code)	0	0	74.57
+ $\mathcal{S}_a$	103	0	74.40
+ $\mathcal{S}_u$ (confidence >0.3)	103	340	74.74
+ $\mathcal{S}_u$ (confidence >0.2)	103	577	75.26
+ $\mathcal{S}_u$ (confidence >0.1)	103	832	**75.60**

Table 4: Performance of NLPROLOG when extracted facts are used as input. Average accuracy over 3 runs is reported. NLPROLOG empowered by 21 natural language explanations and 5 hand-written rules achieves 1% gain in accuracy.

functions. Camburu et al. (2018) extend Stanford Natural Language Inference dataset with NL explanations and demonstrate its usefulness for various goals for training classifiers. Andreas et al. (2016) explores decomposing NL questions into linguistic substructures for learning collections of neural modules which can be assembled into deep networks. Hu et al. (2019) explores using NL instructions as compositional representation of actions for hierarchical decision making. The substructure of an instruction is summarized as a latent plan, which is then executed by another model.

Weakly-supervised learning. Our work is relevant to weakly-supervised learning. Traditional systems use handcrafted rules (Hearst, 1992) or automatically learned rules (Agichtein & Gravano, 2000; Batista et al., 2015) to take a rule-based approach. Hu et al. (2019) incorporate human knowledge into neural networks by using a teacher network to teach the classifier knowledge from rules and train the classifier with labeled data. Li et al. (2018) parse regular expression to get action trees as a classifier that are composed of neural modules, so that essentially training stage is just a process of learning human knowledge. Meanwhile, if we regard those data that are exactly matched by rules as labeled data and the remaining as unlabeled data, we can apply many semi-supervised models such as self learning (Rosenberg et al., 2005), mean-teacher (Tarvainen & Valpola, 2017), and semi-supervised VAE (Xu et al., 2017). However, These models turn out to be ineffective in rule-labeled data or explanation-labeled data due to potentially large difference in label distribution. The data sparsity is also partially solved by distant supervision (Mintz et al., 2009; Surdeanu et al., 2012). They rely on knowledge bases (KBs) to annotate data. However, the methods introduce a lot of noise, which severely hinders the performance. Liu et al. (2017) instead propose to conduct relation extraction using annotations from heterogeneous information source. Again, predicting true labels from noisy sources is challenging.

6 CONCLUSION

In this paper, we presented NMET, a framework that augments sequence classification by exploiting NL explanations as supervision under a low resource setting. We tackled the challenges of modeling the compositionality of NL explanations and dealing with the linguistic variants. Four types of modules were introduced to generalize different type of actions in logical forms, which substantially increases the coverage of NL explanations. A joint training algorithm was proposed to utilize information from both labeled dataset and unlabeled dataset. We conducted extensive experiments on several datasets and proved the effectiveness of our model. Future work includes extending NMET to sequence labeling tasks, and building a cross-domain semantic parser for NL explanations.

REFERENCES

Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, pp. 85–94. ACM, 2000.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 39–48, 2016.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1699–1710, 2015.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

David S Batista, Bruno Martins, and Mário J Silva. Semi-supervised bootstrapping of relationship extractors with distributional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 499–504, 2015.

Oana-Maria Camburu, Tim Rocktäschel, Thomas Lukasiewicz, and Phil Blunsom. e-snli: natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, pp. 9539–9549, 2018.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, 2014.

Braden Hancock, Martin Bringmann, Paroma Varma, Percy Liang, Stephanie Wang, and Christopher Ré. Training classifiers with natural language explanations. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2018, pp. 1884. NIH Public Access, 2018.

Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pp. 539–545. Association for Computational Linguistics, 1992.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pp. 94–99. Association for Computational Linguistics, 2009.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

Hengyuan Hu, Denis Yarats, Qucheng Gong, Yuandong Tian, and Mike Lewis. Hierarchical decision making by generating and following natural language instructions. *arXiv preprint arXiv:1906.00744*, 2019.

Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. 2013.

Shen Li, Hengru Xu, and Zhengdong Lu. Generalize symbolic knowledge with neural rule engine. *arXiv preprint arXiv:1808.10326*, 2018.

Percy Liang, Michael I Jordan, and Dan Klein. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446, 2013.

Hongtao Lin, Jun Yan, Meng Qu, and Xiang Ren. Learning dual retrieval module for semi-supervised relation extraction. In *The Web Conference*, 2019.

Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. Heterogeneous supervision for relation extraction: A representation learning approach. *arXiv preprint arXiv:1707.00166*, 2017.

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

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pp. 1003–1011. Association for Computational Linguistics, 2009.

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. Learning text similarity with siamese recurrent networks. In *Rep4NLP@ACL*, 2016.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of NAACL-HLT*, pp. 528–540, 2018.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems*, pp. 3567–3575, 2016.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. 2005.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 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.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pp. 455–465. Association for Computational Linguistics, 2012.

Antti Tarvainen and Harri Valpola. 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.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 606–615, 2016.

L Weber, P Minervini, J Münchmeyer, U Leser, and T Rocktäschel. Nlprolog: Reasoning with weak unification for question answering in natural language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, ACL 2019, Florence, Italy, Volume 1: Long Papers*, volume 57. ACL (Association for Computational Linguistics), 2019.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302, 2018.

Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. Variational autoencoder for semi-supervised text classification. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1753–1762, 2015.

Luke Zettlemoyer and Michael Collins. Online learning of relaxed ccg grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 678–687, 2007.

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

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 35–45, 2017.

A Appendix

A.1 Predicates

Following Srivastava et al. (2017), we first compile a domain lexicon that maps each word to its syntax and logical predicate. Table 5 lists some frequently used predicates in our parser, descriptions about their function and modules they belong to.

Predicate	Description	Module
Because, Separator ArgX, ArgY, Arg Int, Token, String True, False	Basic conjunction words Subject, object or aspect term in each task Primitive data types Boolean operators	*None*
And, Or, Not, Is, Occur	Logical operators that aggregate matching scores	*Logical Calculation Module*
Left, Right, Between, Within NumberOf	Return True if one string is left/right/between/within some range of the other string Return the number of words in a given range	*Deterministic Function*
AtMost, AtLeast, Direct, MoreThan, LessThan, Equals	Counting (distance) constraints	*Soft Counting Module*
Word, Contains, Link	Return a matching score sequence for a sentence and a query	*String Matching Module*

Table 5: Frequently used predicates

A.2 Examples for collected explanations.

TACRED

Although not a Playboy Playmate , she has appeared in nude pictorials with her Girls Next Door costars and fellow Hefner girlfriends Holly Madison and OBJ-PERSON , then known as SUBJ-PERSON.

```
(Label) per:alternate names
```

```
(Explanation) the term ``then known as'' occurs between
SUBJ-PERSON and OBJ-PERSON and there are no more than six words
between SUBJ-PERSON and OBJ-PERSON.
```

Officials in Mumbai said that the two suspects , David Coleman Headley , an American with links of Pakistan , and SUBJ-PERSON , who was born in Pakistan but is a OBJ-NATIONALITY citizen , both visited Mumbai and several other Indian cities in before the attacks , and may have visited some of the sites that were attacked.

```
(Label) per:origin
```

```
(Explanation) the words ``is a'' appear right before
OBJ-NATIONALITY and the word ``citizen'' is right after
OBJ-NATIONALITY.
```

SemEval 2010 Task 8

The SUBJ-O is caused by the OBJ-O of UV radiation by the oxygen and ozone .

```
(Label) Cause-Effect(e2,e1)
```

```
(Explanation) The phrase ``is caused by the'' occurs between SUBJ
and OBJ and OBJ follows SUBJ.
```

SUBJ-O are parts of the OBJ-O OBJ-O disregarded by the compiler.

```
(Label) Component-Whole(e1,e2)
```

```
(Explanation) The phrase ``are parts of the'' occurs between SUBJ
and OBJ and OBJ follows SUBJ
```

SemEval 2014 Task 4 - restaurant

I am relatively new to the area and tried Pick a bgel on 2nd and was disappointed with the service and I thought the food was overated and on the pricey side. (Term: food)

```
(Label) negative
(Explanation) the words 'overated' is within 2 words after term
```

The decor is vibrant and eye-pleasing with several semi-private boths on the right side of the dining hall, which are great for a date. (Term: decor)

```
(Label) positive
(Explanation) the term is followed by 'vibrant' and 'eye-pleasing'
```

SemEval 2014 Task 4 - laptop

It's priced very reasonable and works very well right out of the box. (Term: works)

```
(Label) positive
(Explanation) the word ``resonable'' occurs before term by no more
than 2 words
```

The DVD drive randomly pops open when it is in my backpack as well, which is annoying. (Term: DVD drive)

```
(Label) negative
(Explanation) The word 'annoying' occurs after term
```

A.3 Implementation Details

We use 300-dimensional word embeddings pre-trained by GloVe (Pennington et al., 2014). The dropout rate for embeddings is 0.96 and the dropout rate for our sentence encoder is 0.5. The hidden state size of the encoder is 300 and the hidden state size of the attention layer is 200. We choose Adagrad as the optimizer and the learning rate for training classifiers is set to 0.5.

For TACRED, in the pretraining stage, we set the learning rate to 0.1. The total epochs for pretraining is 10. The weight for L_{sim} is set to 0.5. The batch size for pretraining is set to 100. For training the classifier, the batch size for labeled data and unlabeled data is 50 and 100 respectively, the weight α for L_u is set to 0.7, the weight β for L_{string} is set to 0.2, the weight γ for L_{sim} is set to 2.5.

For SemEval 2010 Task 8, in the pretraining stage, we set the learning rate to 0.1. The total epochs for pretraining is 10. The weight for L_{sim} is set to 0.5. The batch size for pretraining is set to 10. For training the classifier, the batch size for labeled data and unlabeled data is 50 and 100 respectively, the weight α for L_u is set to 0.5, the weight β for L_{string} is set to 0.1, the weight γ for L_{sim} is set to 2.

For two datasets in SemEval 2014 Task 4, in the pretraining stage, we set the learning rate to 0.5. The total epochs for pretraining is 20. The weight for L_{sim} is set to 5. The batch size for pretraining is set to 20. For training the classifier, the batch size for labeled data and unlabeled data is 10 and 50 respectively, the weight α for L_u is set to 0.5, the weight β for L_{string} is set to 0.1, the weight γ for L_{sim} is set to 2. For ATAE-LSTM, we set hidden state of attention layer to be 300 dimension.

A.4 Full Results

The full results for relation extraction and sentiment analysis are listed in Table 6 and Table 7 respectively.

	TACRED			SemEval		
Metric	Precision	Recall	F1	Precision	Recall	F1
LF ($\mathcal{E}$)	**83.21**	13.56	23.33	**83.19**	21.26	33.86
CBOW-GloVe ($\mathcal{R}+\mathcal{S}$)	28.2±0.7	**44.9±0.9**	34.6±0.4	46.8±1.3	51.2±2.2	48.8±1.1
PCNN ($\mathcal{S}_a$)	43.8±1.6	28.9±1.1	34.8±0.9	51.5±1.9	35.2±1.4	41.8±1.2
PA-LSTM ($\mathcal{S}_a$)	44.4±2.9	38.7±2.2	41.3±0.8	59.9±2.4	54.9±2.2	57.3±1.5
BiLSTM+ATT ($\mathcal{S}_a$)	43.8±2.0	39.4±2.6	41.4±1.0	60.0±2.1	56.2±1.3	58.0±1.6
BiLSTM+ATT ($\mathcal{S}_l$)	42.8±2.6	23.8±2.4	30.4±1.4	54.7±1.0	53.6±1.2	54.1±1.0
Data Programming ($\mathcal{E}+\mathcal{S}$)	45.9±2.8	23.3±2.6	30.8±2.4	51.3±3.5	38.8±4.2	43.9±2.4
Self Training ($\mathcal{S}_a+\mathcal{S}_u$)	45.9±2.3	38.4±2.7	41.7±1.5	57.3±2.1	53.3±0.9	55.2±0.8
Pseudo Labeling ($\mathcal{S}_a+\mathcal{S}_u$)	44.5±1.5	38.9±1.6	41.5±1.2	53.7±2.6	53.4±2.2	53.5±1.2
Mean Teacher ($\mathcal{S}_a+\mathcal{S}_u$)	39.2±1.7	42.6±1.8	40.8±0.9	60.8±1.9	51.9±1.2	56.0±1.1
Mean Teacher ($\mathcal{S}_l+\mathcal{S}_{lu}$)	28.3±5.7	25.4±5.8	25.9±2.2	53.1±3.8	51.6±2.4	52.2±0.7
DualRE ($\mathcal{S}_a+\mathcal{S}_u$)	38.8±4.7	28.6±2.9	32.6±0.7	64.5±0.7	59.2±2.0	61.7±0.9
NMET ($\mathcal{E}+\mathcal{S}$)	49.2±0.9	42.4±1.3	**45.6±0.4**	66.3±1.4	**61.0±2.2**	**63.5±1.0**

Table 6: Full results as supplement to Table 2(a)

	Restaurant			Laptop		
Metric	Precision	Recall	F1	Precision	Recall	F1
LF ($\mathcal{E}$)	**86.5**	4.0	7.7	**90.0**	7.1	13.1
CBOW-GloVe ($\mathcal{R}+\mathcal{S}$)	62.8±2.8	75.3±3.1	68.5±2.9	53.4±1.1	72.6±1.5	61.5±1.3
PCNN ($\mathcal{S}_a$)	67.1±2.1	79.0±1.8	72.6±1.2	53.1±1.0	71.4±1.1	60.9±1.1
ATAE-LSTM ($\mathcal{S}_a$)	65.1±0.4	78.4±0.6	71.1±0.4	49.0±3.1	66.0±4.4	56.2±3.6
ATAE-LSTM ($\mathcal{S}_l$)	65.3±0.5	78.9±0.5	71.4±0.5	48.9±1.5	55.6±2.4	52.0±1.4
Data Programming ($\mathcal{E}+\mathcal{S}$)	65.0±0.0	78.8±0.0	71.2±0.0	53.4±0.1	72.5±0.1	61.5±0.1
Self Training ($\mathcal{S}_a+\mathcal{S}_u$)	65.3±0.7	78.4±0.9	71.2±0.5	50.1±1.8	67.7±2.4	57.6±2.1
Pseudo Labeling ($\mathcal{S}_a+\mathcal{S}_u$)	64.9±0.5	78.0±0.6	70.9±0.4	50.4±1.6	68.4±2.3	58.0±1.9
Mean Teacher ($\mathcal{S}_a+\mathcal{S}_u$)	68.8±2.2	75.7±3.9	72.0±1.5	54.4±1.7	72.3±4.0	62.1±2.3
Mean Teacher ($\mathcal{S}_l+\mathcal{S}_{lu}$)	68.3±0.8	81.0±0.4	74.1±0.4	55.0±4.1	70.3±3.3	61.7±3.7
NMET ($\mathcal{E}+\mathcal{S}$)	69.6±0.9	**83.3±1.8**	**75.8±0.8**	54.6±1.6	**73.9±2.3**	**62.8±1.9**

Table 7: Full results as supplement to Table 2(b)