doi:10.1017/S1471068423000297 First published online 25 July 2023

ASPER: Answer Set Programming Enhanced Neural Network Models for Joint Entity-Relation Extraction[*]

TRUNG HOANG LE, HUIPING CAO and TRAN CAO SON
Department of Computer Science, New Mexico State University, Las Cruces, NM, USA
(*e-mails:* trungle@nmsu.edu, hcao@nmsu.edu, tson@cs.nmsu.edu)

submitted 25 May 2023; accepted 12 June 2023

Abstract

A plethora of approaches have been proposed for joint entity-relation (ER) extraction. Most of these methods largely depend on a large amount of manually annotated training data. However, manual data annotation is time-consuming, labor-intensive, and error-prone. Human beings learn using both data (through induction) and knowledge (through deduction). Answer Set Programming (ASP) has been a widely utilized approach for knowledge representation and reasoning that is elaboration tolerant and adept at reasoning with incomplete information. This paper proposes a new approach, ASP-enhanced Entity-Relation extraction (ASPER), to jointly recognize entities and relations by learning from both data and domain knowledge. In particular, ASPER takes advantage of the factual knowledge (represented as facts in ASP) and derived knowledge (represented as rules in ASP) in the learning process of neural network models. We have conducted experiments on two real datasets and compare our method with three baselines. The results show that our ASPER model consistently outperforms the baselines.

KEYWORDS: joint entity-relation extraction, semi-supervised learning, answer set programming, knowledge-enhanced models

1 Introduction

Entity-relation (ER) extraction is to identify named entities and relations from unstructured text. For joint ER extraction, deep neural network (NN) models have created many successful stories (e.g. the papers by Gupta *et al.* 2016; Eberts and Ulges 2020; Wang and Lu 2020). Despite such success, the supervised NN methods depend on utilizing a large amount of well-labeled training data. However, labeling free text data with entities/relations is time-consuming, labor intensive, and error prone because of a lot of noise, as shown by Chen *et al.* (2022).

Semi-supervised learning (SSL), introduced by Chapelle *et al.* (2006) and Ouali *et al.* (2020), has been utilized to improve predictions by using a small amount of labeled data and a much larger amount of unlabeled data. Among the many SSL approaches, the proxy-label methods described in the papers by Ruder and Plank (2018) and Ouali *et al.* (2020) are one commonly utilized strategy. These approaches create different strategies

[*] Research partially funded by NSF awards #1757207, #1914635, and #1812628.

to utilize the pseudo labels that are predicted from the unlabeled data. However, most of them do not make use of domain knowledge as discussed by Hu *et al.* (2016), which is abundant and very useful, in symbolic forms in the learning process as shown in the survey by Ouali *et al.* (2020).

Recent years have witnessed the increasing interest in utilizing general domain knowledge (e.g. represented as symbolic knowledge) to improve machine learning models to alleviate the issue caused by the lack of large amounts of labeled data. Such efforts include neural symbolic modeling and abductive learning.

Neural symbolic models, referred by us as works that encode the knowledge and rules to be a holistic component of neural network models (e.g. through the design of a new loss function). Such models have achieved great success (e.g. see the papers by Hu *et al.* 2016; d'Avila Garcez *et al.* 2019). However, tightly modeling the symbolic knowledge as a part of NN models suffers from the elaboration tolerant issue where the model is hard to scale to changes of logical representations of facts (e.g. loss functions need to be modified when adding new rules).

Zhou (2019), Dai *et al.* (2019), and Cai *et al.* (2021) introduced abductive learning that combines machine learning models (which are mainly data-driven) and logic programming (which encodes background knowledge and reason about them). In abductive learning, an initial machine learning model M is trained from the labeled data and used to get predicted labels from the unlabeled data (denoted as pseudo labels). Pseudo labels may be wrong or inconsistent when the model M is not effective. Example 3 demonstrates different issues of pseudo labels. The pseudo labels are revised to get a new set of consistent pseudo labels through minimizing the inconsistency of the abduced labels and the knowledge. The revised set of pseudo labels are used to retrain a machine learning model. Most existing abductive learning approaches use first-order logic (FOL) to encode knowledge.

In this work, we propose to design an SSL approach by encoding domain knowledge and rules using Answer Set Programming (ASP) for the joint recognition of entities and relations. The purpose of using the domain knowledge is to generate consistent pseudo labels (consistent w.r.t. the knowledge base) and derive more pseudo labels that cannot be predicted using the pure data-driven models. ASP instead of FOL is used because of multiple advantages ASP provides. ASP is a simple, rule-based, and declarative language that possesses several theoretical building block results which support the development of provably correct programs. In addition, ASP is non-monotonic, which is important for dealing with commonsense knowledge, and supports an elaboration tolerant development of programs. For non-logical experts, ASP-rules are easier to use and to understand than FOL-formulae.

The main contributions of this work are as follows.

- A new framework, ASP-enhanced Entity-Relation extraction (ASPER), is introduced to make use of sophisticated domain knowledge in neural network models. ASP-encoded knowledge and rules intend to generate higher quality pseudo labels, which are further used to improve the model. As far as we know, this is the first work that incorporates logic programming to deep learning models for joint ER extraction.

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press

- ASPER introduces novel commonsense rules to select pseudo labels that may improve the model performance with higher probabilities.
- The experimental evaluation on two real datasets shows that the proposed ASPER consistently improves the other baselines.

In what follows, Section 2 reviews the related work, Section 3 formally defines the problem and related notations, Section 4 explains our new framework, Section 5 shows our experimental results, and Section 6 concludes the work.

2 Related work

SSL methods are designed to make use of a small amount of labeled data and a much larger amount of unlabeled data in the learning process. Traditional SSL methods including self-training and tri-training have been revisited recently by Ruder and Plank (2018). Both self-training and tri-training utilize an iterative process to improve the initial model(s) trained on the small amount of labeled data by iteratively adding pseudo labels to the training set. In each iteration, self-training picks pseudo labels that have higher prediction probability and tri-training picks the pseudo labels that are agreed by at least two models. A surprising finding in such revisit is that the classic tri-training, introduced in the paper by Zhou and Li (2005), strategy with minor changes can outperform many recent NN models.

Many recent SSL approaches have been proposed to conduct ER extraction. Hu *et al.* (2021) proposes a self-training based SSL method for relation extraction. In each iteration, it adopts meta-learning to generate higher quality pseudo labels. Curriculum labeling, introduced in the paper by Cascante-Bonilla *et al.* (2021), borrows the idea of curriculum learning discussed in the paper by Bengio *et al.* (2009), which uses easy samples first and proceeds to use hard samples, and proposes a self-paced curriculum (curriculum labeling) in the pseudo labeling process. However, this model does not use any symbolic knowledge. Most of these approaches do not make use of domain knowledge in symbolic forms.

Similar to SSL, neural network models also alleviate the issue of insufficient labeled data. Hu *et al.* (2016) is the first work of integrating logic rules with NN models. It proposes an iterative distillation method to enhance several NNs with declarative first-order logic (FOL) rules, which is encoded using soft logic. NeurASP, introduced in the paper by Yang *et al.* (2020), also employs ASP and neural networks. Its knowledge base is predefined and all the atoms are utilized in the learning process. However, our knowledge base is used to generate answer sets with consistent pseudo labels and some answer sets (when multiple are available) may not be utilized in the learning process.

Our work also shares similarity with the framework of abductive learning such as those described in the papers Zhou (2019), Huang *et al.* (2020) in that symbolic knowledge is used to improve the quality of pseudo labels. However, our work is different from abductive learning in several aspects. Abductive learning (e.g. the paper by Huang *et al.* 2020) derives abduced pseudo labels through an optimization process. Once these labels are revised, they are used to retrain a model. When they retrain a model, all the pseudo labels are utilized. Our approach utilizes a subset of the pseudo labels, which have higher probability to be true, to retrain a model. In addition, the pseudo labels are iteratively refined using ASP, which provides powerful reasoning capability.

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press

3 Problem definition and terminology

This section defines our research problem and related terminology.

Problem definition: Given a dataset D_L (training data) with labeled entities and relations, a dataset D_{UL} without any annotated labels, where $|D_L| \ll |D_{UL}|$, and a knowledge base[1] KB which encodes the common sense facts and reasoning rules, our problem is to learn an NN model $M \leftarrow f(D_L, D_{UL}, KB)$ to capture the hidden patterns about entities and relations in the datasets $D_L \cup D_{UL}$.

Definition 1 (Pseudo labels)**.** *Given a model M and a dataset D_{UL} without any annotated labels, the predicted labels from D_{UL} using M are called pseudo labels.*

The possible entity and relation labels that occur in the training data D_L are represented as ent and rel. Given any token or word in a sentence, we use b and e to represent the beginning and ending locations of that token in the sentence. The b and e are in the range of 0 and the total number of tokens of a sentence. An entity pseudo label is in the form of

$$ent(b, e), conf \tag{1}$$

It means that the tokens at the locations $[b,\ e-1]$ in the sentence is of entity type ent. Here, $conf$ is a value in the range of [0,1] indicating the confidence of the prediction. A relation pseudo label is in the form of

$$rel(b, e, b', e'), conf \tag{2}$$

Here, b (b') and e (e') represent the beginning and ending locations of the first (second) token in the relation. To make the descriptions more intuitive, we sometimes represent a relation as

$$rel(tokens, tokens'), conf \tag{3}$$

where $tokens$ ($tokens'$) are the first (second) tokens at locations $[b,\ e)$ (and $[b',\ e')$). Without loss of generality, we may omit $conf$ when writing the entity and relation pseudo labels.

Example 1 (Running example notations)**.** *In later examples, we will use some well defined entity types. For example, org, loc, and $peop$ represent organization, location, and people respectively. Some predefined relations are $livedIn$, $locatedIn$, and $orgBasedIn$. They describe one person, a location, and an organization lives in, is located in, or is based in a location respectively.*

4 ASP-enhanced neural network models for Entity-Relation extraction (ASPER)

This section presents our proposed ASPER method. ASPER targets at utilizing answer sets and ASP to improve the quality of pseudo labels and derive more useful labels to retrain the model.

[1] The KB is domain-dependent. We discuss practical ideas on developing such KB for the joint ER extraction problem in the later section.

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press

4.1 ASPER framework

The framework of ASPER is shown in Algorithm 1. ASPER first trains an initial model using the limited amount of training data (Line 1) and improves the model through an iterative process using ASP revised pseudo labels (Lines 3–20).

To train an initial neural network model, we utilize the SpERT architecture proposed in the paper by Eberts and Ulges (2020) due to its lightweight nature. Multiple iterations (Steps 3–20) are used to improve the model. In each iteration, ASPER predicts the entities and relations in a sentence x (i.e. the pseudo labels) using the model M (Line 7) where M can be the initial model (trained in Line 1) or the retrained model (Line 18). Then, it utilizes ASP to update these pseudo labels (Lines 8–10). The updated pseudo labels coupled with the selected sentences are then used to retrain the model (Lines 12–18). There are many ways to revise pseudo labels. We define a preference relation over the sets of revised labels (answer sets) based on the notion of the probability of a set of revised labels (Definition 2 in Section 4.2). This preference relation is then used to select the most preferred set of revised labels. The iteration condition can be that the prediction of the unlabeled data does not change anymore or the number of iterations reaches a threshold. In our experiments, we set the number of iterations to be a fixed number.

4.1.1 Generate pseudo labels

Using the initially trained model or a model that is trained in the previous iteration M, the algorithm can recognize the entities and relations in the unlabeled dataset D_{UL}. The predicted entity and relation pseudo labels are in the form of equations (1) and (2).

Algorithm 1 ASPER framework

```
Input: Labeled data $D_L$; Unlabeled data $D_{UL}$; Knowledge Base $KB$
Parameter: Confidence step parameter $\Delta$ (e.g. 20)
Output: The model $M$
 1: Learn an initial model $M$ from $D_L$
 2: $\Delta_t = 100 - \Delta$
 3: while iteration condition is not met do
 4:     $\mathbb{Z} \leftarrow \emptyset$ # The set of the selected answer sets for all the sentences in $D_{UL}$
 5:     $D_{aug} \leftarrow \emptyset$ # The pseudo labels augmented to train the model
 6:     for each sentence $x \in D_{UL}$ do
 7:         $z \leftarrow$ GenPseudoLabels($M$, $x$)
 8:         $A_S \leftarrow$ Convert2Atoms($z$)
 9:         $W \leftarrow$ ReviseUsingASP($A_S \cup KB$) #$W$ is an answer set associated with a confidence value
            $W.conf$
10:         $\mathbb{Z} \leftarrow \mathbb{Z} \cup W$
11:     end for
12:     $T \leftarrow$ the confidence value at $\Delta_t$ percentile of all the answer sets in $\mathbb{Z}$
13:     for each answer set $W \in \mathbb{Z}$ do
14:         if $W.conf \geq T$ then
15:             $D_{aug} \leftarrow D_{aug} \cup W$
16:         end if
17:     end for
18:     Train model $M$ on $D_L \cup D_{aug}$ from scratch
19:     $\Delta_t = \Delta_t - \Delta$
20: end while
21: return $M$
```

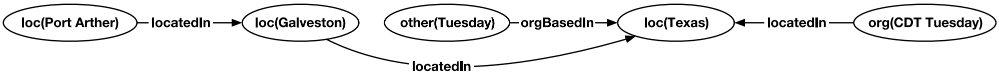


Fig. 1. Example of pseudo labels (a node represents an entity and a directed edge between a source (first tokens) and a destination (second tokens) represents a relation).

Example 2 (Pseudo labels)**.** *Given a sentence "CDT Tuesday is in the area of Port Arther and Galveston, Texas." the predicted pseudo labels look like the following:*

org(0,2),	*0.888*	*other(1,2),*	*0.799*	*locatedIn(7,9,10,11),*	*0.998*
loc(7,9),	*0.998*	*loc(10,11),*	*0.998*	*locatedIn(0,2,12, 13),*	*0.993*
loc(12,13),	*0.998*	*orgBasedIn(1,2,12,13),*	*0.777*	*locatedIn(10,11,12,13),*	*0.993*

The pseudo label "org(0,2), 0.888" means that the token from location 0 to 1 (which is "CDT Tuesday") is an organization (org). This prediction has confidence value 0.888. Similarly, the pseudo label "loc(7,9)" means that "Port Arther" (the tokens at locations 7 and 8) is of type location (loc). Correspondingly, the predicted relation "locatedIn(7,9,10,11)" means that "Port Arther" is located in "Galveston." The other entities and relations can be interpreted accordingly. To assist the understanding of the relations, we create a figure (Figure 1) for these entities and relations.

4.1.2 Improve pseudo label quality using answer sets

The predicted entity and relation pseudo labels may be wrong (just like any machine learning model does) or inconsistent.

Example 3 (Inconsistent and hidden pseudo labels)**.** *Example 2 shows the pseudo labels predicted from one sentence. They have different types of inconsistencies.*

- *(inconsistent labels) The relation* $locatedIn(CDT\ Tuesday, Texas)$ *and entity* $org(CDT\ Tuesday)$ *are not consistent because the first term of* $locatedIn$ *needs to be a location, but* $CDT\ Tuesday$ *is an organization. Similarly, the entity* $other(Tuesday)$ *and relation* $orgBasedIn(Tuesday, Texas)$ *are not consistent because the first term of* $orgBasedIn$ *needs to be an organization.*
- *(overlapping entities) The two entities* $org(0,2)$ *and* $other(1,2)$ *overlap because "Tuesday" is a part of "CDT Tuesday." It makes more sense not to have both.*
- *(hidden labels) Given* $locatedIn(Port\ Arther, Galveston)$ *and* $locatedIn(Galveston, Texas)$*, another relation* $locatedIn(Port\ Arther, Texas)$ *should be valid. However, the model does not predict this relation.*

In utilizing pseudo labels to improve a model, a recognized problem is *confirmation bias* as mentioned in the paper by Cascante-Bonilla *et al.* (2021), which means that utilizing the wrong pseudo labels to retrain a model can amplify the error. It is critical to control which pseudo labels are utilized to retrain the model.

The issues listed above are not inclusive. Their root issue is the ineffective pure data-driven model learned from insufficient training data. The most commonly identified issue with an ineffective model is its inaccurate predictions. We target at addressing the root issue by somehow correcting the wrongly predicted pseudo labels. Our ASPER framework

(Lines 8–10) improves the quality of pseudo labels by computing a consistent set of pseudo labels (an answer set). It first converts all the pseudo labels (z) to a set of atoms A_S that ASP can process (using Function *Convert2Atoms*). Given A_S and the knowledge base KB, there may be multiple answer sets. *ReviseUsingASP* utilizes the rules in the KB to calculate a probability for each answer set and chooses the one with the highest probability and associates with it a confidence level. The details of the two steps are described in Section 4.2.2.

4.1.3 Model retraining with improved pseudo labels

Once we get the improved pseudo labels from the unlabeled dataset D_{UL}, these improved pseudo labels are put to $\mathbb{Z}$ (Line 10) and are used to help retrain the model.

We observe that some answer sets have much higher confidence values than others. The pseudo labels in these answer sets tend to be correct with higher probabilities. Based on this observation, the model retraining first utilizes the pseudo labels in the answer sets with higher confidence values and proceeds to use pseudo labels in answer sets with lower confidence values. This idea is the same as that in curriculum labeling proposed by Cascante-Bonilla *et al.* (2021) that uses a portion (with high prediction confidence) of the pseudo labels to retrain a model in each iteration. This curriculum idea is implemented through the use of Δ_t in Line 12. With the iterations proceed, Δ_t decreases (Line 19). At the end, when Δ_t becomes zero, the model retraining uses the pseudo labels in all the answer sets.

4.2 Computing improved pseudo labels via ASP

Background. ASP, proposed in the papers by Marek and Truszczyński (1999), and Niemelä (1999), is a knowledge representation and reasoning (KR&R) approach to problem solving using logic programs under answer set semantics introduced by Gelfond and Lifschitz (1990). In ASP, a problem is solved by first encoding it as an ASP program, whose answer sets correspond one-to-one to the solutions of the problem. The encoded ASP program is then given to an ASP solver (e.g. `clingo` as described in the paper by Gebser *et al.* 2014) to compute answer sets and solutions can be extracted from these answer sets.

The ASP language also includes language-level extensions to simplify the use of ASP in practical applications. We will make use of the *choice atom* of the form $l\,\{l_1;\ldots;l_n\}\,u$ where l_i is an atom, l and u are integers. Intuitively, it says that the number of literal l_i that is true must be within a lower bound l and an upper bound u. If l (resp. u) is not specified, then the lower (resp. upper) bound is 0 (resp. $+\infty$) by default. A choice atom can appear in the head or body of a rule or after the default negation.

4.2.1 ASP for computing pseudo labels

First, the pseudo labels representing the entities (equation (1)) and relations (equation (2)) are represented in ASP using atoms of the form (4) or (5).

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press

$$atom(entity(ent, b, e), conf) \tag{4}$$

$$atom(relation(rel, b, e, b', e'), conf) \tag{5}$$

Let A_S be the collection of atoms of the form (4) or (5) for a sentence S. A sentence is a part of a dataset D that often has declarative knowledge associated with it. In this paper, we consider the different types of knowledge that are usually available given D which can be specified as follows:

1. *Type declaration*: a type declaration defines the type of a relation and is given in the form

$$type\_def(rel, ent, ent'). \tag{6}$$

A type declaration by equation (6) says that relation rel is between entities ent and ent'. For example, in the domain `CoNLL04` (see next section), the relation $liveIn$ is specified by the atom $type\_def(liveIn, peop, loc)$ which says that it is a relationship between entities of the types $peop$ and loc.

2. *Inference rule*: in many domains, there are relationships among relations. For example, $locatedIn$ is transitive in the sense that if area A is located in area B and B is located in C then A is located in C. This rule can easily be specified by an ASP rule of the following form[2]:

$$rule(X, Y, Z) \leftarrow Body \tag{7}$$

where X, Y, and Z is of the form $relation(R, B, E, B', E')$ and $Body$ is domain-specific information. The rule relating to $locatedIn$ discussed above can be encoded as follows:

$$\begin{aligned}
&rule(relation(locatedIn, B_1, E_1, B_2, E_2), relation(orgbasedIn, B_o, E_o, B_1, E_1),\\
&\qquad \underline{relation(orgbasedIn, B_o, E_o, B_2, E_2)}) \leftarrow\\
&atom(relation(locatedIn, B_1, E_1, B_2, E_2)), atom(relation(orgbasedIn, B_o, E_o, B_1, E_1)),\\
&not\ atom(relation(orgbasedIn, B_o, E_o, B_2, E_2)).
\end{aligned}$$

The head of the above ASP rule encodes an inference rule, whose first two labels (the relations on the first line) are predicted but the third relation (underlined) is missing in the set of predicted labels. This inference rule is used for inferring the third pseudo label only if the first two pseudo labels exist and the third pseudo label is not in the predicted model.

3. *Optional parameters*: in some dataset, entity pseudo labels cannot overlap and/or each sentence has at least one relation. Such information can be specified by setting different flags. In our experiments, we use the following:

$$overlap\_fl. \qquad \%\ \text{true if present} \tag{8}$$

$$relation\_fl. \qquad \%\ \text{true if present} \tag{9}$$

Form (8) forbids overlapping entities while (9) signals that each sentence should have a relation.

[2] We use variables (strings starting with an uppercase letter) in the logic program. A rule with variables is the collection of ground rules obtained from substituting variables with any possible values; in this case, variables refer to locations in the sentence.

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press

4. *Other rules*: any ASP rule can be a part of the domain-dependent part. Preferably, these rules work with the predicates defined below.

We refer to the collection of rules of the form (7)–(9) and any other rules for a domain D as KB_D. We denote that A_S is *inconsistent* when it contains pseudo labels that either contradict to each other or violate the rules in KB_D.

We next describe Π, the ASP program that takes the domain-dependent knowledge KB_D and the set of pseudo labels for a sentence A_S as inputs and produces a consistent set of pseudo labels. Π uses the following main predicates:

- $pi(X)$: the prediction of X (entity/relation pseudo label) might be incorrect (pi stands for $possible\_incorrect$);
- $ok(X)$: the prediction of X is considered as correct; as such, X can be used as pseudo label (for further training);
- $nok(X)$: X is not included in the output (set of pseudo labels); and
- $inf(X)$: X is derived using domain-dependent rules.

1. *Overlap checking rules*: Π contains the following rules:

$$\begin{aligned} 2\{pi(entity(N,B,E)); pi(entity(N',B',E'))\} &\leftarrow overlap\_fl,\\ atom(entity(N,B,E)), B < B', E > B'&, atom(entity(N',B',E')). \quad (10)\\ \leftarrow overlap\_fl, ok(entity(N,B,E)), B < B', E > B'&, ok(entity(N',B',E')). \quad (11)\end{aligned}$$

Rule (10) states that if the starting location B' of an entity (N') lies within the interval of the some other entity (N) then the prediction of the two entities might be wrong which is represented by the predicate pi. This leads to the constraints (11) that prevents the consideration of both entities at the same time when they overlap in a sentence. The presence of $overlap\_fl$ in these rules indicates that they are in effect only if $overlap\_fl$ is set to true in KB_D. Similar rules and constraints are also implemented for the case that the starting indices of both entities are the same. They are omitted for brevity to save space and are listed in the appendix.

2. *Type checking rules*: This type of rules is used to make sure that the entity types and relation types are consistent with the type declaration information in KB_D:

$$\begin{aligned} 2\{pi(relation(R,B,E,B',E')); pi(entity(N,B,E))\} &\leftarrow type\_def(R,N_1,N_2),\\ atom(relation(R,B,E,B',E')), atom(entity(N,B,E))&, N_1 \neq N. \quad (12)\\ \leftarrow ok(relation(R,B,E,\_,\_)), ok(entity(N,B,E))&, type\_def(R,N',\_), N \neq N'. (13)\end{aligned}$$

Rule (12) states that if the first entity in a predicted relation is different from its specified type then both the predicted relation and the entity might be wrong. Constraint (13) disallows the acceptance of both the relation and entity if their types do not march the specification. Again, we omit some rule relating to the second entity and the relation.

3. *Type inference rules*: These rules use the type declarations from KB_D to infer the type of accepted entities or the possible incorrect predictions of relations and entities.

$$\begin{aligned} 2\{ok(entity(N,B,E)); ok(entity(N',B',E'))\} &\leftarrow\\ type\_def(R,N,N'), ok(relation(R,B,E,B',E'))&. \quad (14)\\ pi(entity(N',B',E')) &\leftarrow atom(relation(R,B,E,B',E')),\\ pi(entity(N,B,E)), type\_def(R,N,N')&. \quad (15)\end{aligned}$$

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press

Rule (14) says that if a relation R is accepted as a true prediction then we should also accept its first and second entity with the type specified by the type declaration of R. Rule (15) indicates that if the first entity of a relation R is potentially incorrect then so is its second entity.

4. *Rules for inference from predictions*: The following rules are used for processing an inference rule specified in KB_D via atom of the form $rule(X, Y, Z)$.

$$6\{pi(X); pi(Y); pi(Z); inf(Z); dependency(X, Z); dependency(Y, Z)\} \leftarrow \\ rule(X, Y, Z), atom(X), atom(Y),\ not\ atom(Z). \tag{16}$$

$$\leftarrow ok(Y), inf(Y), dependency(X, Y), not\ ok(X). \tag{17}$$

$$\leftarrow rule(X, Y, Z), ok(X), ok(Y),\ not\ ok(Z). \tag{18}$$

Rule (16) says that if we have an inference rule $rule(X, Y, Z)$ and X and Y were predicted but not Z then Z is an inferred prediction and all the predictions might be incorrect. Furthermore, Z depends on X and Y. Constraint (17) states that if Y is an inferred atom and depends on X then the acceptance of Y cannot be done separately from the acceptance of X. Constraint (18), on the other hand, states that if $rule(X, Y, Z)$ is an inference rule then the acceptance of X and Y cannot be done separately from the acceptance of Z.

5. *Rules for checking the existence of relation*: The following rule records the acceptance of some relation pseudo label whenever $relation\_fl$ is defined:

$$relation\_exists \leftarrow ok(relation(R, \_, \_, \_, \_)), relation\_fl. \tag{19}$$

6. *Rules for selection of a consistent set of pseudo labels*: This set of rules defines the various types of atoms that will be used for computing the probability of a set of pseudo labels and selecting a set of pseudo labels.

$$atom(X) \leftarrow atom(X, P). \tag{20}$$

$$prob(X, P) \leftarrow atom(X, P), ok(X). \tag{21}$$

$$ok(X) \leftarrow atom(X),\ not\ pi(X). \tag{22}$$

$$invprob(X, P) \leftarrow atom(X, P), nok(X). \tag{23}$$

$$\{ok(X)\} \leftarrow 1\{atom(X); inf(X)\}, pi(X). \tag{24}$$

$$nok(X) \leftarrow 1\{atom(X); inf(X)\},\ not\ ok(X). \tag{25}$$

Rule (20) projects an input $atom(X, P)$ to define $atom(X)$ for use with other part of the program. Rules (21)–(23) define the predicates *prob* and *invprob* that are used in computing the probability of the set of selected labels (see later). Rule (22) says that if there is no information about the potential incorrectness of the prediction of X then X must be accepted as a pseudo label. Rule (24) states that if the prediction of X might be incorrect then X could be accepted or not accepted as a pseudo label. Rule (25) says that if X is not selected as a pseudo label then $nok(X)$ is true.

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press

In summary, Π is the collection of rules of the Rules (10)–(25). Together with the set A_S of the predicted labels and the knowledge base KB_D, we can compute a new set of pseudo labels by computing the answer sets of $\pi(S) = A_S \cup KB_D \cup \Pi$. Each answer set W of $\pi(S)$ consists of a set of atoms of the form $ok(X)$. We say that $L(W) = \{X \mid ok(X) \in W\}$ is the set of pseudo labels corresponding to W. Intuitively, $L(W)$ encodes a revision of A_S. For an arbitrary sentence S in a domain D, we can prove several properties of the program $\pi(S)$ such as (i) $\pi(S)$ is consistent; (ii) if A_S does not contain any relation then $relation\_exists$ does not belong to any answer set of $\pi(S)$; and (iii) for every W, $L(W)$ does not contain any overlapping pseudo labels if $overlap\_fl$ is true; and $L(W)$ does not contain type-inconsistent relations or entities. Intuitively, (i) represents the fact that there is always some revision of A_S. This can be proven using the splitting theorem proposed by Lifschitz and Turner (1994); (ii) holds because of Rule (19); and (iii) holds due to the rules and constraints defined for type inference and checking. Due to space limitation, we omit the formal proof. The next example illustrates the use of Π.

Example 4. *Consider the sentence in Example 2. The program produces twenty answer sets (the $KB_{\texttt{CoNLL04}}$ and the answer sets of the program are given in the Appendix). We observe that*

- *Some answer set does not contain any atom of the form $ok(relation(\ldots))$;*
- *No answer set contains both $ok(entity(org, 0, 2))$ and $ok(entity(other, 1, 2))$ because they overlap each other and $KB_{\texttt{CoNLL04}}$ contains $overlap\_fl$;*
- *$ok(relation(orgBasedIn, 1, 2, 12, 13))$ belongs to some answer sets but not all. It is because $entity(other, 1, 2)$ is of the type other that does not match the required type (org) in the definition of $orgbasedIn$; and*
- *If $ok(relation(locatedIn, 7, 9, 10, 11))$ and $ok(relation(locatedIn, 10, 11, 12, 13))$ belong to an answer set then $ok(relation(locatedIn, 7, 9, 12, 13))$ also belongs to the answer set due to the transitivity of $locatedIn$, a part of the $KB_{\texttt{CoNLL04}}$.*

Note that encoding knowledge in ASP incurs extra works. However, compared with manually labeling a large amount of data, this extra works pay off.

4.2.2 Computing the most preferred answer set

Definition 2 (Preference score of an answer set). *Given an answer set W, its preference score is defined as*

$$pref(W) = \Pi_{prob(a,p) \in W}\, p * \Pi_{invprob(a,p) \in W} (1 - p) \tag{26}$$

The preference score $pref(W)$ is the product of two terms, the first term is the product of the confidence level p of every pseudo label a such that $ok(a) \in W$ (hence, $prob(a, p) \in W$, due to Rule (21)) and the second term is the product of the complement confidence level $1-p$ of pseudo label a such that $ok(a) \notin W$ (hence, $invprob(a, p) \in W$, due to Rule (23)). It is easy to see that for two answer sets A and B such that $L(B) \subseteq L(A)$ (i.e. A contains all acceptable labels in B), if $prob(l, p) \in A$ and $p \geq 0.5$ for every $l \in L(A) \setminus L(B)$, then $p(A) \geq p(B)$. Intuitively, this probability definition favors answer sets containing more (w.r.t. $\subseteq$) pseudo labels whose confidence level is greater than 0.5. When $relation\_fl$ is set to true, we set $pref(W) = 0$ if $relation\_exists \notin W$. Preference will be given to the answer set with maximal probability. When all answer sets have zero preference, we prefer

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press

those with higher number of entity pseudo labels. The selection of the most preferred answer set is implemented using `clingo` and its PythonAPI library. The confidence level of an answer set W (i.e. $W.conf$) is defined by $\min\{p \mid prob(l,p) \in W\}$.

5 Experiments

The algorithms and models are implemented in Python 3.8 and run on a server with two 32-core GPU @3.9 GHz, 512 GB RAM, and one NVIDIA A100 GPU. The Clingo version is 5.5.2.

Data. We use two datasets, CoNLL04 (Eberts and Ulges 2020; Roth and Yih 2004; Gupta *et al*. 2016; Wang and Lu 2020) and SciERC (Eberts and Ulges 2020; Luan *et al*. 2018), which have been utilized in other entity/relation extraction work. The CoNLL04 dataset contains sentences extracted from newspapers. We employ the training (922 sentences), development (231 sentences) and test set (288 sentences) split, which is similar to that of Gupta *et al*. (2016). The SciERC dataset consists of abstracts of artificial intelligence research papers. The training/development/test split is 1861/275/551 sentences, which is the same as that of Luan *et al*. (2018).

These datasets all have training sets. To utilize these datasets to verify the effectiveness of our method, we do not use the whole training set to train the initial model. Instead, we use a small percentage of the training data (e.g. $p_{tr} \in (0,1]$) as the actual training data D_L and use the remaining $(1-p_{tr})$ training data as the unlabeled data D_{UL} to calculate pseudo labels. The original testing data is still utilized to test the model performance. To get stable results, for each dataset, we randomly choose five subsets (each one contains p_{tr} of the training data) from the training data and train five models. Then, we report the averaged results from the five models.

Performance metric. We report the micro and macro F_1 values for entities (E), relations (R), and relations together with entities (ER). The micro F_1 is calculated globally by counting the total true positives (TP), false negatives (FN), and false positives (FP). In all the counting, a special class (not-an-entity or not-a-relation), which is encoded as zero, is never treated as positive. For example, considering the E type, all the correctly predicted non-zero entities are counted as TP. Among the wrongly predicted entities, an entity is counted as FP if it is wrongly predicted to be non-zero, and an entity is counted as FN if its true class is non-zero. Some wrongly predicted entities are counted in both FP and FN. The macro F_1 is obtained by calculating the prediction F_1 when treating each class as positive (and all the others as negative) and averaging the F_1s for all the classes. We also report the running time of the models.

Methods. We compare our ASPER method with three classical and state-of-the-art baselines listed below. (1) *Self-training* as described in the papers by McClosky *et al*. (2006), Reichart and Rappoport (2007). For this method, we use 90% as the threshold to select pseudo labels to be included for model retraining. (2) *Curriculum labeling (CL)*: this method retrains the model using the curriculum-labeling strategy proposed by Cascante-Bonilla *et al*. (2021). This is a state-of-the-art approach for SSL using pseudo labels. It has one hyper parameter (stepping threshold) controlling the confidence value of pseudo labels that are included in the training of the model in one iteration. This parameter is set to the same (20%) as the original paper. (3) *Tri-training*: this method retrains the model using the tri-training strategy proposed by Zhou and Li (2005). A re-

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press

Table 1. *Performance comparison of ASPER and other baselines on the two datasets (E: entity, R: relation, ER: entity and relation; $p_{tr} = 10\%$)*

	CoNLL04 dataset					
	F_1 (micro)			F_1 (macro)		
Method	E	R	ER	E	R	ER
Self-train	77.74±1.7	41.76±5.7	41.39±5.7	72.50±1.9	43.19±6.0	42.82±6.0
CL	77.49±1.1	41.61±3.0	41.35±3.2	72.03±1.6	43.07±3.8	42.77±4.0
Tri-train	78.63±2.4	42.60±6.7	42.29±6.7	72.49±2.5	42.99±7.1	42.64±7.2
ASPER	**81.25**±1.2	**52.47**±3.6	**52.41**±3.6	**75.90**±1.7	**53.32**±4.0	**53.27**±4.0

	SciERC dataset					
	F_1 (micro)			F_1 (macro)		
Method	E	R	ER	E	R	ER
Self-train	56.72±1.2	18.60±2.6	12.36±1.7	54.43±1.4	11.07±3.7	6.98±2.3
CL	60.75±0.8	31.00±2.1	20.81±1.0	59.19±0.4	22.00±3.8	15.55±1.8
Tri-train	**60.99**±0.7	27.43±1.9	18.94±1.4	**59.52**±0.4	17.09±3.6	11.59±2.7
ASPER	60.34±0.6	**32.30**±1.2	**21.73**±1.2	59.10±0.4	**22.72**±3.1	**16.06**±2.3

cent study by Ruder and Plank (2018) has shown that the classic tri-training method is still a strong baseline for neural semi-supervised learning for natural language processing.

For fair comparison, we run five iterations (retraining of the model) for every model. For our model, Δ is set to be 20% as that in curriculum-labeling approach.

5.1 Effectiveness analysis

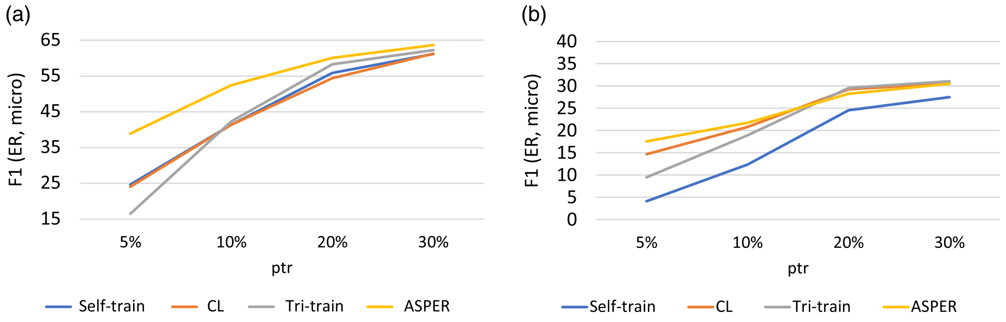
We conduct experiments to examine the effectiveness of our approach. Our first set of experiments is to compare our ASPER method with the other baselines. In these experiment, p_{tr} is set to be 10%. That is, 10% of the training data forms D_L. Table 1 shows the results on the CoNLL04 and the SciERC datasets. It shows that ASPER outperforms all the other baselines on all the calculated F_1 measurement on recognizing relations (R) and both entities and relations (ER) no matter it is at the micro or macro level. For Entity (E) recognition, Tri-training is slightly better than our method. This is because our training process gives higher preferences to sentences with potentially correct relations. These results show the superiority of our proposed method.

When more training data is available and the KB cannot provide extra information than what the labeled data can provide, ASPER may not beat the pure data-driven models such as tri-training and curriculum labeling. However, ASPER is able to improve (at least not hurt) its base deep learning model (the SpERT model in this paper) that ASPER is built upon no matter whether the KB can provide much more information than the training data or not.

We conduct a more detailed analysis about the running of ASPER by showing its performance in three iterations (other iterations show similar trend). This analysis is to show the quality of the pseudo label revision. The quality can be measured by comparing

Table 2. *The effect of the ASPER algorithm to improve the quality of the pseudo labels on the two datasets. (E: entity, R: relation, ER: entity and relation; $p_{tr} = 10\%$)*

		CoNLL04 dataset					
		F_1 (micro)			F_1 (macro)		
ASPER		E	R	ER	E	R	ER
Iter 1	no-ASP	75.65	41.62	41.07	69.81	42.86	42.29
	with ASP	77.06	44.45	44.45	71.16	45.40	45.40
Iter 2	no-ASP	78.68	49.78	49.08	73.24	50.86	50.16
	with ASP	79.01	50.19	50.19	73.49	51.27	51.27
Iter 3	no-ASP	79.98	52.92	52.66	74.25	53.97	53.72
	with ASP	80.24	53.62	53.62	74.52	54.67	54.67
		SciERC dataset					
		F_1 (micro)			F_1 (macro)		
ASPER		E	R	ER	E	R	ER
Iter 1	no-ASP	60.34	30.87	21.98	58.78	21.47	16.59
	with ASP	60.34	31.12	22.09	58.78	21.75	16.79
Iter 2	no-ASP	60.86	32.59	23.10	59.39	22.34	17.25
	with ASP	60.86	32.83	23.24	59.39	22.58	17.42
Iter 3	no-ASP	60.65	33.11	23.53	59.31	22.79	17.74
	with ASP	60.65	33.12	23.54	59.31	22.82	17.76



CoNLL04 SciERC

Fig. 2. Performance comparison (varying training data amount).

the pseudo labels and the ground truth labels (which are the training data with the correct labels, but are not directly used for training) and calculating the F_1 score.

Table 2 shows the detailed analysis of how the ASP component helps improve the quality of the generated pseudo labels. We can see that after each iteration, the F1 of the ASP generated pseudo labels is always higher than that of the raw pseudo labels. This confirms that the use of answer sets and ASP helps improve the quality of the pseudo labels. Table 2 also shows that the performance improvement in earlier iterations is better than that in later iterations. This is also consistent with our design of utilizing curriculum labeling, that is, answer sets with higher confidence values are used in earlier iterations (Section 4.1.3).

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press

Table 3. *Ablation study on ASPER; CoNLL04*

	(F_1 micro)			F_1 (macro)		
	E	R	ER	E	R	ER
with all rules	**81.25**	**52.47**	**52.41**	**75.90**	**53.32**	**53.27**
with all rules except the *relation_exists* rule	76.64	34.13	33.84	70.56	34.87	34.57
without any rules	76.74	31.07	31.07	70.52	31.99	31.99

The second set of experiments examines the effect of the amount of initial training data on ASPER. We change the amount of initial training dataset D_L by varying the percentage p_{tr} with different values (5%, 10%, 20%, 30%).

The results are reported in Figure 2. We only report the F1 of relation with entity (micro) because the trend of the other F1 values is the same. The figure shows that, when there is less initial training data, the overall performance is worse. However the positive effect of ASPER is more obvious when there are less training data, which can be observed from the larger gap between ASPER and other methods for smaller p_{tr} (e.g. 5%). This result is consistent with our intuition of designing ASPER to alleviate the issue of insufficient amount of training data. Figure 2(b) also shows that our method does outperform, but has comparable performance as, CL and tri-training when p_{tr} is larger. The major reason is that the knowledge base is less effective in capturing the characteristics of the second domain (research articles). More effective rules need to be developed to enrich the knowledge base in the future.

The third set of experiments conduct an ablation study to investigate the effect of the rules in the ASP program. Due to space limitation, we present this analysis on one dataset. Table 3 shows the results. The first row (*with all rules*) shows the results when all the rules are utilized. The third row (*without any rules*) on the other hand shows the results when no rule is utilized. The results (improvement of the first row comparing to the third row) clearly demonstrate that the rules contribute positively to the performance of ASPER. We conduct a further analysis about the effect of the different type of rules and find that the *relation_exists* rule (Rule (19)) plays the most significant role. The second row shows the results from the program while the *relation_exists* rule is not utilized, but all the other rules are used. The improvement of all the other rules to the algorithm (which is captured by the difference between the 2nd and the 3rd row) is not as much as the *relation_exists* rule (which is observed from the difference between the 1st and the 2nd row).

5.2 Efficiency analysis

We also examine the running time of the different methods to understand the overhead brought by the ASP program. Due to space constraint, we report the summarized data here. Self-training, curriculum labeling, and ASPER use similar amount of time. On the CoNLL04 dataset, it takes approximately 40–50 min to run the five iterations. Tri-training's time is approximately three times of the other three methods because it needs to train three models in each iteration. The overhead of using ASP to generate the

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press

updated pseudo labels is about 30 s in each iteration. This time is negligible compared with the expensive NN model training.

6 Conclusions

In this paper, we presented a novel method ASPER, which leverages Answer Set Programming (ASP) to improve the performance of Neural Network models in the joint recognition of entities and relations from text data when limited amount of training data is available. ASPER makes use of pseudo labels. The ASP program encodes different types of commonsense rules by taking advantage of the commonsense domain knowledge. The experiments on two real datasets show that ASPER can report significantly better results than the other baselines in most cases.

References

Bengio, Y., Louradour, J., Collobert, R. and Weston, J. 2009. Curriculum learning. In *ICML*, vol. 382. ACM, 41–48.

Cai, L., Dai, W., Huang, Y., Li, Y., Muggleton, S. H. and Jiang, Y. 2021. Abductive learning with ground knowledge base. In *IJCAI*. ijcai.org, 1815–1821.

Cascante-Bonilla, P., Tan, F., Qi, Y. and Ordonez, V. 2021. Curriculum labeling: Revisiting pseudo-labeling for semi-supervised learning. In *AAAI*. AAAI Press, 6912–6920.

Chapelle, O., Schölkopf, B. and Zien, A., Eds. 2006. *Semi-Supervised Learning*. The MIT Press.

Chen, M., Zhao, Y., He, B., Han, Z., Wu, B. and Yao, J. 2022. Learning with noisy labels over imbalanced subpopulations. CoRR abs/2211.08722.

Dai, W., Xu, Q., Yu, Y. and Zhou, Z. 2019. Bridging machine learning and logical reasoning by abductive learning. In *NeurIPS*, 2811–2822.

d'Avila Garcez, A. S., Gori, M., Lamb, L. C., Serafini, L., Spranger, M. and Tran, S. N. 2019. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *FLAP 6,* 4, 611–632.

Eberts, M. and Ulges, A. 2020. Span-based joint entity and relation extraction with transformer pre-training. In *ECAI 2020 - 24th European Conference on Artificial Intelligence*. Frontiers in Artificial Intelligence and Applications, vol. 325. IOS Press, 2006–2013.

Gebser, M., Kaminski, R., Kaufmann, B. and Schaub, T. 2014. Clingo = ASP + control: Preliminary report. In *Technical Communications of the 13th ICLP*, vol. 14(4-5).

Gelfond, M. and Lifschitz, V. 1990. Logic programs with classical negation. In *Logic Programming: Proc. of the Seventh International Conference*, D. Warren and P. Szeredi, Eds., 579–597.

Gupta, P., Schütze, H. and Andrassy, B. 2016. Table filling multi-task recurrent neural network for joint entity and relation extraction. In *COLING*. ACL, 2537–2547.

Hu, X., Zhang, C., Ma, F., Liu, C., Wen, L. and Yu, P. S. 2021. Semi-supervised relation extraction via incremental meta self-training. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics, 487–496.

Hu, Z., Ma, X., Liu, Z., Hovy, E. H. and Xing, E. P. 2016. Harnessing deep neural networks with logic rules. In *ACL*.

Huang, Y., Dai, W., Yang, J., Cai, L., Cheng, S., Huang, R., Li, Y. and Zhou, Z. 2020. Semi-supervised abductive learning and its application to theft judicial sentencing. In *20th IEEE International Conference on Data Mining, ICDM 2020*. IEEE, 1070–1075.

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press

Lifschitz, V. and Turner, H. 1994. Splitting a logic program. In *Proceedings of the Eleventh International Conference on Logic Programming*, P. Van Hentenryck, Ed., 23–38.

Luan, Y., He, L., Ostendorf, M. and Hajishirzi, H. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *EMNLP*, 3219–3232.

Marek, V. and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: A 25-year Perspective*, 375–398.

McClosky, D., Charniak, E. and Johnson, M. 2006. Effective self-training for parsing. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings*. The Association for Computational Linguistics.

Niemelä, I. 1999. Logic programming with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence 25,* 3,4, 241–273.

Ouali, Y., Hudelot, C. and Tami, M. 2020. An overview of deep semi-supervised learning. CoRR abs/2006.05278.

Reichart, R. and Rappoport, A. 2007. Self-training for enhancement and domain adaptation of statistical parsers trained on small datasets. In *ACL*.

Roth, D. and Yih, W. 2004. A linear programming formulation for global inference in natural language tasks. In *CoNLL*. ACL, 1–8.

Ruder, S. and Plank, B. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *ACL 2018*. Association for Computational Linguistics, 1044–1054.

Wang, J. and Lu, W. 2020. Two are better than one: Joint entity and relation extraction with table-sequence encoders. In *(EMNLP)*, 1706–1721.

Yang, Z., Ishay, A. and Lee, J. 2020. Neurasp: Embracing neural networks into answer set programming. In *IJCAI 2020*, C. Bessiere, Ed. ijcai.org, 1755–1762.

Zhou, Z. 2019. Abductive learning: Towards bridging machine learning and logical reasoning. *Science China Information Sciences 62,* 7, 76101:1–76101:3.

Zhou, Z. and Li, M. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering 17,* 11, 1529–1541.

https://doi.org/10.1017/S1471068423000297 Published online by Cambridge University Press