Learning from Explicit and Implicit Supervision Jointly For Algebra Word Problems

Shyam Upadhyay[1] **Ming-Wei Chang**[2] **Kai-Wei Chang**[3] **Wen-tau Yih**[2]
[1]University of Illinois at Urbana-Champaign, Urbana, IL, USA
[2]Microsoft Research, Redmond, WA, USA
[3]University of Virginia, Charlottesville, VA, USA

Abstract

Automatically solving algebra word problems has raised considerable interest recently. Existing state-of-the-art approaches mainly rely on learning from human annotated equations. In this paper, we demonstrate that it is possible to efficiently mine algebra problems and their numerical solutions with little to no manual effort. To leverage the mined dataset, we propose a novel structured-output learning algorithm that aims to learn from both explicit (e.g., equations) and implicit (e.g., solutions) supervision signals *jointly*. Enabled by this new algorithm, our model gains 4.6% absolute improvement in accuracy on the ALG-514 benchmark compared to the one without using implicit supervision. The final model also outperforms the current state-of-the-art approach by 3%.

1 Introduction

Algebra word problems express mathematical relationships via narratives set in a real-world scenario, such as the one below:

Maria is now four times as old as Kate.
Four years ago, Maria was six times as
old as Kate. Find their ages now.

The desired output is an *equation system* which expresses the mathematical relationship symbolically: $m = 4 \times n$ and $m - 4 = 6 \times (n - 4)$ where m and n represent the age of Maria and Kate, respectively. The *solution* (i.e., $m = 40, n = 10$) can be found by a *mathematical engine* given the equation systems. Building efficient automatic algebra word problem solvers have clear values for online education scenarios. The challenge itself also provides a good test bed for evaluating an intelligent agent that understands natural languages, a direction advocated by artificial intelligence researchers (Clark and Etzioni, 2016).

One key challenge of solving algebra word problems is the lack of fully annotated data (i.e., the annotated equation system associated with each problem). In contrast to annotating problems with binary or categorical labels, manually solving algebra word problems to provide correct equations is time consuming. As a result, existing benchmark datasets are small, limiting the performance of supervised learning approaches. However, thousands of algebra word problems have been posted and discussed in online forums, where the solutions can be easily mined, despite the fact that some of them could be incorrect. It is thus interesting to ask whether a better algebra problem solver can be learned by leveraging these *noisy* and *implicit* supervision signals, namely the solutions.

In this work, we address the technical difficulty of leveraging implicit supervision in learning an algebra word problem solver. We argue that the effective strategy is to learn from both explicit and implicit supervision signals *jointly*. In particular, we design a novel online learning algorithm based on structured-output Perceptron. By taking both kinds of training signals together as input, the algorithm iteratively improves the model, while at the same time it uses the intermediate model to find candidate equation systems for problems with only numerical solutions.

Our contributions are summarized as follows.

- We propose a novel learning algorithm (Section 3 and 4) that jointly learns from both explicit and implicit supervision. Under different settings, the proposed algorithm outperforms the existing supervised and weakly supervised algorithms (Section 6) for algebra word problems.
- We mine the problem-solution pairs for algebra word problems from an online forum and show that we can effectively obtain the implicit supervision with little to no manual effort (Section 5).[1]
- By leveraging both implicit and explicit supervision signals, our final solver outperforms the state-of-the-art system by 3% on ALG-514, a popular benchmark data set proposed by (Kushman et al., 2014).

2 Related Work

Automatically solving mathematical reasoning problems expressed in natural language has been a long-studied problem (Bobrow, 1964; Newell et al., 1959; Mukherjee and Garain, 2008). Recently, Kushman et al. (2014) created a template-base search procedure to map word problems into equations. Then, several following papers studied different aspects of the task: Hosseini et al. (2014) focused on improving the generalization ability of the solvers by leveraging extra annotations; Roy and Roth (2015) focused on how to solve arithmetic problems without using any pre-defined template. In (Shi et al., 2015), the authors focused on number word problems and proposed a system that is created using semi-automatically generated rules. In Zhou et al. (2015), the authors simplified the inference procedure and pushed the state-of-the-art benchmark accuracy. The idea of learning from implicit supervision is discussed in (Kushman et al., 2014; Zhou et al., 2015; Koncel-Kedziorski et al., 2015), where the authors train the algebra solvers using only the solutions with little or no annoated equation systems. We discuss this in detail in Section 4.

Solving automatic algebra word problems can be viewed as a semantic parsing task. In the semantic parsing community, the technique of learning from implicit supervision signals has been applied (under the name *response-driven learning* (Clarke et al., 2010)) to knowledge base question answering tasks such as Geoquery (Zelle and Mooney, 1996) and WebQuestions (Berant et al., 2013) or mapping instructions to actions (Artzi and Zettlemoyer, 2013). In these tasks, researchers have shown that it is possible to train a semantic parser only from question-answer pairs, such as "*What is the largest state bordering Texas?*" and "*New Mexico*" (Clarke et al., 2010; Liang et al., 2013; Yih et al., 2015).

One key reason that such implicit supervision is effective is because the correct semantic parses of the questions can often be found using the answers and the knowledge base alone, with the help of heuristics developed for the specific domain. For instance, when the question is relatively simple and does not have complex compositional structure, paths in the knowledge graph that connect the answers and the entities in the narrative can be interpreted as legitimate semantic parses. However, as we will show in our experiments, learning from implicit supervision alone is not a viable strategy for algebra word problems. Compared to the knowledge base question answering problems, one key difference is that a large number (potentially infinitely many) of different equation systems could end up having the same solutions. Without a database or special rules for combining variables and coefficients, the number of candidate equation systems cannot be trimmed effectively, given only the solutions.

From the algorithmic point of view, our proposed learning framework is related to several lines of work. Similar efforts have been made to develop latent structured prediction models (Yu and Joachims, 2009; Chang et al., 2013; Zettlemoyer and Collins, 2007) to find latent semantic structures which best explain the answer given the question. Our algorithm is also influenced by the discriminative reranking algorithms (Collins, 2000; Ge and Mooney, 2006; Charniak and Johnson, 2005) and models for learning from intractable supervision (Steinhardt and Liang, 2015).

Recently, Huang et al. (2016) collected a large

[1]The new resource and the dataset we used for training is available soon on `https://aka.ms/dataimplicit` and `https://aka.ms/datadraw`

number of noisily annotated word problems from online forums. While they collected a large-scale dataset, unlike our work, they did not demonstrate how to utilize the newly crawled dataset to improve existing systems. It will be interesting to see if our proposed algorithm can make further improvements using their newly collected dataset.[2]

3 Problem Definition

Table 1 lists all the symbols representing the components in the process. The input algebra word problem is denoted by $\mathbf{x}$, and the output $\mathbf{y} = (T, A)$ is called a *derivation*, which consists of an *equation system template* T and an *alignment* A. A template T is a family of equation systems parameterized by a set of coefficients $\mathcal{C}(T) = \{c_i\}_{i=1}^k$, where each coefficient c_i aligns to a textual number (e.g., *four*) in a word problem. Let $\mathcal{Q}(x)$ be all the *textual numbers* in the problem $\mathbf{x}$, and $\mathcal{C}(T)$ be the coefficients to be determined in the template T. An alignment is a set of tuples $A = \{(q, c) \mid q \in \mathcal{Q}(x), c \in \mathcal{C}(T) \cup \{\epsilon\}\}$, where the tuple (q, ϵ) indicates that the number q is not relevant to the final equation system. By specifying the value of each coefficient, it identifies an equation system belonging to the family represented by template T. Together, T and A generate a complete equation system, and the solution $\mathbf{z}$ can be derived by the mathematical engine E.

Following (Kushman et al., 2014; Zhou et al., 2015), our strategy of mapping a word problem to an equation system is to first choose a template that consists of variables and coefficients, and then align each coefficient to a textual number mentioned in the problem. We formulate the mapping between an algebra word problem and an equation system as a structured learning problem. The output space is the set of all possible derivations using templates that are observed in the training data. Our model maps $\mathbf{x}$ to $\mathbf{y} = (T, A)$ by a linear scoring function $\mathbf{w}^T \Phi(\mathbf{x}, \mathbf{y})$, where $\mathbf{w}$ is the model parameters and Φ is the feature functions. At test time, our model scores all the derivation candidates and picks the best one according to the model score. We often refer to $\mathbf{y}$ as a semantic parse, as it represents the semantics of the algebra word problem.

[2]The dataset has not been made public at the time of publication.

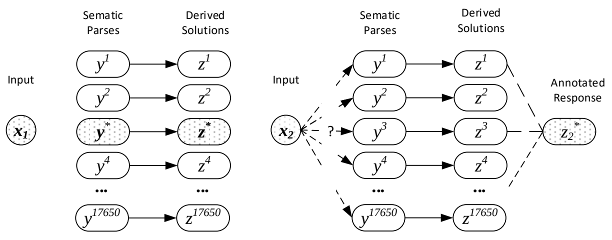

Figure 1: **Left:** Explicit supervision signals. Note that the solution $\mathbf{z}$ can be derived by the semantic parses $\mathbf{y}$. **Right:** Implicit supervision signals. In this case, we only have the annotated response $\mathbf{z}_2^*$. It is difficult to use $\mathbf{z}_2^*$ to find the correct derivation, as multiple derivations may lead to the same solution. Therefore, the learning algorithm has to explore the output space to guide the model in order to match the annotated response.

Properties of Implicit Supervision Signals We discuss some key properties of the implicit supervision signal to explain several design choices of our algorithm. Figure 1 illustrates the main differences between implicit and explicit supervision signals.

Algorithms that learn from implicit supervision signals face the following challenges. First, the learning system usually does not model directly the correlations between the input $\mathbf{x}$ and the solution $\mathbf{z}$. Instead, the mapping is handled by an external procedure such as a mathematical engine. Therefore, $E(\mathbf{y})$ is effectively a one-directional function. As a result, finding semantic parses (derivations) from responses (solutions) $E^{-1}(\mathbf{z})$ can sometimes be very slow or even intractable. Second, in many cases, even if we could find a semantic parse from responses, multiple combinations of templates and alignments could end up with the same solution set (e.g., the solutions of equations $2 + x = 4$ and $2 \times x = 4$ are the same). Therefore, the implicit supervision signals may be incomplete and noisy, and using the solutions alone to guide the training procedure might not be sufficient. Finally, since we need to have a complete derivation before we can observe the response of the mathematical engine E, we cannot design efficient inference methods such as dynamic programming algorithms based on partial feedback. As a result, we have to perform exploration during learning to search for fully constructed semantic parses that can generate the correct solution.

Term	Symbol	Example
Word Problem	$\mathbf{x}$	*Maria is now four times as old as Kate. Four years ago, Maria was six times as old as Kate. Find their ages now.*
Derivation (Semantic Parse)	$\mathbf{y} = (T, A)$	$(\{m - a \times n = -1 \times a \times b + b, m - c \times n = 0\}, A)$
Solution	$\mathbf{z}$	$n = 10, m = 40$
Mathematical Engine	$E : \mathbf{y} \rightarrow \mathbf{z}$	After determining the coefficients, the equation system is $\{m = 4 \times n, m - 4 = 6 \times (n - 4)\}$. The solution is thus $n = 10$, $m = 40$.
Variables	v	m, n
Textual Number[3]	$\mathcal{Q}(x)$	{four, Four, six}
Equation System Template	T	$\{m - a \times n = -1 \times a \times b + b, m - c \times n = 0\}$
Coefficients	$\mathcal{C}(T)$	a, b, c
Alignment	A	six $\rightarrow a$, Four $\rightarrow b$, four $\rightarrow c$

Table 1: Notation used in this paper to formally describe the problem of mapping algebra word problems to equations.

4 Learning from Mixed Supervision

We assume that we have two sets: $D_e = \{(\mathbf{x}_e, \mathbf{y}_e)\}$ and $D_m = \{(\mathbf{x}_m, \mathbf{z}_m)\}$. D_e contains the fully annotated equation system $\mathbf{y}_e$ for each algebra word problem $\mathbf{x}_e$, whereas in D_m, we have access to the numerical solution $\mathbf{z}_m$ to each problem, but not the equation system ($\mathbf{y}_m = \emptyset$). We refer to D_e as the *explicit set* and D_m as the *implicit set*. For the sake of simplicity, we explain our approach by modifying the training procedure of the structured Perceptron algorithm (Collins, 2002).[4]

As discussed in Section 3, the key challenge of learning from implicit supervision is that the mapping $E(\mathbf{y})$ is one-directional. Therefore, the correct equation system cannot be easily derived from the numerical solution. Intuitively, for data with only implicit supervision, we can explore the structure space Y and find the best possible derivation $\tilde{\mathbf{y}} \in Y$ according to the current model. If $E(\tilde{\mathbf{y}})$ matches $\mathbf{z}$, then we can update the model based on $\tilde{\mathbf{y}}$. Following this intuition, we propose *MixedSP* (Algorithm 1).

For each example, we use an approximate search algorithm to collect top scoring candidate structures. The algorithm first ranks the top-K templates according to the model score, and forms a candidate set by expanding all possible derivations that use the K templates (Line 3). The final candidate set is $\Omega = \{\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^K\} \subset Y$.

When the explicit supervision is available (i.e., $(\mathbf{x}_i, \mathbf{y}_i) \in D_e$), our algorithm follows the standard structured prediction update procedure. We find the best scoring structure $\hat{\mathbf{y}}$ in Ω and then update the model using the difference of the feature vectors between the gold output structure $\mathbf{y}_i$ and the best scoring structure $\hat{\mathbf{y}}$ (Line 6).

When only implicit supervision is available (i.e., $(\mathbf{x}_i, \mathbf{z}_i) \in D_m$), our algorithm uses the current model to conduct a guided exploration, which iteratively finds structures that best explain the implicit supervision, and use the explanatory structure for making updates. As mentioned in Section 3, we have to explore and examine each structure in the candidate set Ω. This is due the fact that partial structure cannot be used for finding the right response, as getting response $E(\mathbf{y})$ requires complete derivations. In Line 9, we want to find the derivations $\mathbf{y}$ where its solution $E(\mathbf{y})$ matches the implicit supervision $\mathbf{z}_i$. More specifically,

$$\tilde{\mathbf{y}} = \arg\min_{\mathbf{y} \in \Omega} \Delta(E(\mathbf{y}), \mathbf{z}_i), \qquad (1)$$

where Δ is a loss function to estimate the disagreement between $E(\mathbf{y})$ and $\mathbf{z}_i$. In our experiments, we simply set $\Delta(E(\mathbf{y}), \mathbf{z}_i)$ to be 0 if the solution partially matches, and 1 otherwise.[5] If more than one derivation achieves the minimal value of $\Delta(E(\mathbf{y}), \mathbf{z}_i)$, we break ties by choosing the derivation with higher score $\mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y})$. This tie-

[4]Our approach can be easily extended to other structured learning algorithms such as Structured SVM (Taskar et al., 2004; Tsochantaridis et al., 2004).

[5]The mined solutions are often incomplete for some variables (e.g. solution `y=6` but no value for `x` could be mined). We allow partial matches so that the model can learn from the incomplete implicit signals as well.

Algorithm 1 Structured Perceptron with Mixed Supervision. (**MixedSP**)

```
Input: $D_e, D_m, L = |D_e| + |D_m|, T, K, \gamma \in [0, 1)$
 1: for $t = 1 \dots N$ do                              ▷ training epochs
 2:     for $i = 1 \dots L$ do
 3:         $\Omega \leftarrow$ find top-K structures $\{\mathbf{y}\}$ approximately
 4:         if $\mathbf{y}_i \neq \emptyset$ then            ▷ explicit supervision
 5:             $\hat{\mathbf{y}} \leftarrow \arg\max_{\mathbf{y} \in \Omega} \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y})$
 6:             $\mathbf{w} \leftarrow \mathbf{w} + \eta\,(\phi(\mathbf{x}, \mathbf{y}_i) - \phi(\mathbf{x}, \hat{\mathbf{y}}))$
 7:         else if $t \geq \gamma N$ then            ▷ implicit supervision
 8:             $\hat{\mathbf{y}} \leftarrow \arg\max_{\mathbf{y} \in \Omega} \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y})$
 9:             Pick $\tilde{\mathbf{y}}$ from $\Omega$ by Eq. (1).  ▷ exploration
10:             $\mathbf{w} \leftarrow \mathbf{w} + \eta\,(\phi(\mathbf{x}, \tilde{\mathbf{y}}) - \phi(\mathbf{x}, \hat{\mathbf{y}}))$
11: Return the average of $\mathbf{w}$
```

breaking strategy is important – in practice, several derivations may lead to the gold numerical solution; however, only few of them are correct. The tie-breaking strategy relies on the current model and the structured features $\phi(\mathbf{x}_i, \mathbf{y})$ to filter out incorrect derivations during training. Finally, the model is updated using $\tilde{\mathbf{y}}$ in Line 10.

Similar to curriculum learning (Bengio et al., 2009), it is important to control when the algorithm starts exploring the output space using weak supervision. Exploring too early may mislead the model, as the structured feature weights $\mathbf{w}$ may not be able to help filter out incorrect derivations, while exploring too late may lead to under-utilization of the implicit supervision. We use the parameter γ to control when the model starts to learn from implicit supervision signals. The parameter γ denotes the fraction of the training time that the model uses purely explicit supervision.

Key Properties of Our Algorithm The idea of using solutions to train algebra word problem solvers has been discussed in (Kushman et al., 2014) and (Zhou et al., 2015). However, their implicit supervision signals are created from clean, fully supervised data, and the experiments use little to no explicit supervision examples.[6] While their algorithms are interesting, the experimental setting is somewhat unrealistic as the implicit signals are simulated.

On the other hand, the goal of our algorithm is to significantly improve a strong solver with a large quantity of unlabeled data. Moreover, our implicit supervision signals are noisier given that we crawled the data automatically, and the clean labeled equation systems are not available to us. As a result, we have made several design choices to address issues of learning from noisy implicit supervision signals in practice.

First, the algorithm is designed to perform updates *conservatively*. Indeed, in Line 10, the algorithm will not perform an update if the model could not find any parses matching the implicit signals in Line 9. That is, if $\Delta(E(\mathbf{y}), \mathbf{z}_i) = 1$ for all $\mathbf{y} \in \Omega$, $\tilde{\mathbf{y}} = \hat{\mathbf{y}}$ due to the tie-breaking mechanism. This ensures that the algorithm drives the learning using only those structures which lead to the correct solution, avoiding undesirable effects of noise.

Second, the algorithm does *not* use implicit supervision signals in the early stage of model training. Learning only on clean and explicit supervision helps derive a better intermediate model, which later allows exploring the output space more efficiently using the implicit supervision signals.

Existing semantic parsing algorithms typically use either implicit or explicit supervision signals *exclusively* (Zettlemoyer and Collins, 2007; Berant et al., 2013; Artzi and Zettlemoyer, 2013). In contrast, *MixedSP* makes use of both explicit and implicit supervised examples mixed at the training time.

5 Mining Implicit Supervision Signals

In this section, we describe the process of collecting SOL-2K, a data set containing question-solution pairs of algebra word problems from a Web forum[7], where students and tutors interact to solve math problems.

A word problem posted on the forum is often accompanied by a detailed explanation provided by tutors, which includes a list of the relevant equations. However, these posted equations are not suitable for direct use as labeled data, as they are often imprecise or incomplete. For instance, tutors often omit many simplification steps when writing the equations. A commonly observed example is that `(5-3)x+2y` would be directly written as `2x+2y`. Despite being

[6]Prior work (Kushman et al., 2014) has used only 5 explicit supervision examples when training with solutions.

[7]`http://www.algebra.com`

mathematically equivalent, learning from the latter equation is not desirable as the model may learn that 5 and 3 appearing the text are irrelevant. An extreme case of this is when tutors directly post the solution (such as `x=2` and `y=5`), without writing any equations. Another observation is that tutors often write two-variable equation systems with only one variable. For example, instead of writing `x+y=10, x-y=2`, many tutors pre-compute `x=10-y` using the first equation and substitute it in the second one, which results in `10-y-y=2`. It is also possible that the tutor wrote the incorrect equation system, but while explaining the steps, made corrections to get the right answer. These practical issues make it difficult to use the crawled equations for explicit supervision directly.

On the other hand, it is relatively easy to obtain question-solution pairs with simple heuristics. We use a simple strategy to generate the solution from the extracted equations. We greedily select equations in a top-down manner, declaring success as soon as we find an equation system that can be solved by a mathematical engine (we used SymPy (Sympy Development Team, 2016)). Equations that cause an exception in the solver (due to improper extraction) are rejected. Note that the solution thus found may be incorrect (making the mined supervision noisy), as the equation system used by the solver may contain an incorrect equation. To ensure the quality of the mined supervision, we use several simple rules to further filter the problems. For example, we remove questions that have more than 15 numbers. We found that usually such questions were not a single word problem, but instead concatenations of several problems.

Note that our approach relies only on a few rules and a mathematical engine to generate (noisy) implicit supervision from crawled problems, with no human involvement. Once the solutions are generated, we discarded the equation systems used to obtain them. Using this procedure, we collected 2,039 question-solution pairs. For example, the solution to the following mined problem was "6" (The correct solutions are 6 and 12.):

> Roz is twice as old as Grace. In 5 years the sum of their ages will be 28. How old are they now?

Settings	Explicit sets		Implicit sets
	(D_e)		(D_m)
Dataset	ALG-514	DRAW-1K	SOL-2K
# temp.	24	224	Unknown
# prob.	514	1,000	2,039
Vocab.	1.83k	2.2k	6.8k

Table 2: The statistics of the data sets.

6 Experiments

In this section, we demonstrate the effectiveness of the proposed approach and empirically verify the design choices of the algorithm. We show that our joint learning approach leverages mined implicit supervision effectively, improving system performance without using additional manual annotations (Section 6.1). We also compare our approach to existing methods under different supervision settings (Section 6.2).

Experimental Settings Table 2 shows the statistics of the datasets. The ALG-514 dataset (Kushman et al., 2014) consists of 514 algebra word problems, ranging over a variety of narrative scenarios (object counting, simple interest, etc.). Although it is a popular benchmark for evaluating algebra word solvers, ALG-514 has only 24 templates. To test the generality of different approaches, we thus conduct experiments on a newly released data set, DRAW-1K[8] (Upadhyay and Chang, 2016), which covers more than 200 templates and contains 1,000 algebra word problems. The data is split into training, development, and test sets, with 600/200/200 examples, respectively.

The SOL-2K dataset contains the word problem-solution pairs we mined from online forum (see Section 5). Unlike ALG-514 and DRAW-1K, there are no annotated equation systems in this dataset, and only the solutions are available. Also, no preprocessing or cleaning is performed, so the problem descriptions might contain some irrelevant phrases such as "please help me". Since all the datasets are generated from online forums, we carefully examined and removed problems from SOL-2K that are identical to problems in ALG-514 and DRAW-1K, to ensure fairness. We set the number of iterations

[8]`https://aka.ms/datadraw`

to 15 and the learning rate η to be 1.

For all experiments, we report *solution accuracy* (whether the solution was correct). Following Kushman et al. (2014), we ignore the ordering of answers when calculating the solution accuracy. We report the 5-fold cross validation accuracy on ALG-514 in order to have a fair comparison with previous work. For DRAW-1K, we report the results on the test set. In all the experiments, we only use the templates that appear in the corresponding explicit supervision.

Following (Zhou et al., 2015), we do not model the alignments between noun phrases and variables. We use a similar set of features introduced in (Zhou et al., 2015), except that our solver does not use rich NLP features from dependency parsing or coreference-resolution systems. We follow (Kushman et al., 2014) and set the beam-size K to 10, unless stated otherwise.

6.1 Joint Learning from Mixed Supervision

Supervision Protocols We compare the following training protocols:

- *Explicit* ($D = \{(\mathbf{x}_e, \mathbf{y}_e)\}$): the standard setting, where fully annotated examples are used to train the model (we use the structured Perceptron algorithm as our training algorithm here).

- *Implicit* ($D = \{(\mathbf{x}_m, \mathbf{z}_m))\}$): the model is trained on SOL-2K dataset only (i.e., only implicit supervision). This setting is similar to the one in (Liang et al., 2013; Clarke et al., 2010).

- *Pseudo* ($D = \{(\mathbf{x}_m, \tilde{Z}^{-1}(\mathbf{z}_m, \mathbf{x}_m))\}$): where we use $\tilde{Z}^{-1}(\mathbf{z}, \mathbf{x})$ to denote a pseudo derivation whose solutions match the mined solutions. Similar to the approach in (Yih et al., 2015) for question answering, here we attempts to recover (possibly incorrect) explicit supervision from the implicit supervision by finding parses whose solution matches the mined solution. For each word problem, we generated a pseudo derivation $\tilde{Z}^{-1}(\mathbf{z}, \mathbf{x})$ by finding the equation systems whose solutions that match the mined solutions. We conduct a brute force search to find $\tilde{Z}^{-1}(\mathbf{z}, \mathbf{x})$ by enumerating all possible derivations. Note that this process can be very slow for datasets like DRAW-1K because the brute-force search needs to examine more than 200 templates for each word problem. Ties are broken by random.

- *E+P* ($D = \{(\mathbf{x}_e, \mathbf{y}_e)\} \cup \{(\mathbf{x}_m, \tilde{Z}^{-1}(\mathbf{z}_m, \mathbf{x}_m))\}$): a baseline approach that jointly learns by combining the dataset generated by *Pseudo* with the *Explicit* supervision.

- *MixedSP* ($D = \{(\mathbf{x}_e, \mathbf{y}_e)\} \cup \{(\mathbf{x}_m, \mathbf{z}_m))\}$): the setting used by our proposed algorithm. The algorithm trained the word problem solver using both explicit and implicit supervision jointly. We set the parameter γ to 0.5 unless otherwise stated. In other words, the first half of the training iterations use only explicit supervision.

Note that *Explicit*, *E+P*, and *MixedSP* use the same amount of labeled equations, although *E+P* and *MixedSP* use additional implicit supervised resources.

Results Table 3 lists the main results. With implicit supervision from mined question-solution pairs, *MixedSP* outperforms *Explicit* by around 4.5% on both datasets. This verifies the claim that the joint learning approach can benefit from the noisy implicit supervision. Note that with the same amount of supervision signals, *E+P* performs poorly and even under-performs *Explicit*. The reason is that the derived derivations in SOL-2K can be noisy. Indeed, we found that about 70% of the problems in the implicit set have more than one template that can produce a derivation which matches the mined solutions. Therefore, the pseudo derivation selected by the system might be wrong, even if they generate the correct answers. As a result, *E+P* can commit to the possibly incorrect pseudo derivations before training, and suffer from error propagation. In contrast, *MixedSP* does not commit to a derivation and allows the model to choose the one best explaining the implicit signals as training progresses.

As expected, using only the implicit set D_m performs poorly. The reason is that in both *Implicit* and *Pseudo* settings, the algorithm needs to select one from many derivations that match the labeled solutions, and use the selected derivation to update the model. When there are no explicit supervision

Dataset	D_e	D_m		D_e and D_m	
	Expl.	Pseudo	Impl.	E+P	MixedSP
ALG-514	78.4	54.1	63.7	73.3	**83.0**
DRAW-1K	55.0	33.5	39.0	48.5	**59.5**

Table 3: The solution accuracies of different protocols on ALG-514 and DRAW-1K.

signals, the model can use incorrect derivations to update the model. As a result, models on both *Implicit* and *Pseudo* settings perform significantly worse than the *Explicit* baseline in both datasets, even if the size of SOL-2K is larger than the fully supervised data.

6.2 Comparisons to Previous Work

We now compare to previous approaches for solving algebra word problems, both in fully supervised and weakly supervised settings.

Comparisons of Overall Systems We first compare our systems to the systems that use the same level of explicit supervision (fully labeled examples). The comparison between our system and existing systems are in Fig 2a and 2b. Compared to previous systems that were trained only on explicit signals, our *Explicit* baseline is quite competitive. On ALG-514, the accuracy of our baseline system is 78.4%, which is 1.3% lower than the best reported accuracy achieved by the system ZDC15 (Zhou et al., 2015). We suspect that this is due to the richer feature set used by ZDC15, which includes features based on POS tags, coreference and dependency parses, whereas our system only uses features based on POS tags. Our system is also the best system on DRAW-1K, and performs much better than the system KAZB14 (Kushman et al., 2014). Note that we could not run the system ZDC15 on DRAW-1K because it can only handle limited types of equation systems. Although the *Explicit* baseline is strong, the *MixedSP* algorithm is still able to improve the solver significantly through noisy implicit supervision signals without using manual annotation of equation systems.

Comparisons of Weakly Supervised Algorithms In the above comparisons, *MixedSP* benefits from the mined implicit supervision as well as using Algorithm 1. Since there are several practical limitations for us to run previously proposed weakly supervised algorithms in our settings, in the following, we perform a direct comparison between *MixedSP* and existing algorithms in their corresponding settings. Note that the implicit supervision in weak supervision settings proposed in earlier work is noise-free, as it was simulated by hiding equation systems of a manually annotated dataset.

Zhou et al. (2015) proposed a weak supervision setting where the system was provided with the set of all templates, as well as the solutions of all problems during training. Under this setting, they reported 72.3% accuracy on ALG-514. Note that such high accuracy can be achieved mainly because that the complete and correct templates were supplied.

In this setting, running the *MixedSP* algorithm is equivalent to using the *Implicit* setting with clean implicit supervision signals. Surprisingly, *MixedSP* can obtain 74.3% accuracy, surpassing the weakly supervised model in (Zhou et al., 2015) on ALG-514. Compared to the results in Table 3, note that when using noisy implicit signals, it cannot obtain the same level of results, even though we had more training problems (2,000 mined problems instead of 514 problems). This shows that working with real, noisy weak supervision is much more challenging than working on simulated, noise-free, weak supervision.

Kushman et al. (2014) proposed another weak supervision setting (5EQ+ANS in the paper), in which explicit supervision is provided for only 5 problems in the training data. For the rest of problems, only their solutions are provided. The 5 problems are chosen such that their templates constitute the 5 most common templates in the dataset. This weak supervision setting is harder than that of (Zhou et al., 2015), as the solver only has the templates for 5 problems, instead of the templates for all problems. Under this setting, our *MixedSP* algorithm achieves 53.8%, which is better than 46.1% reported in (Kushman et al., 2014).

6.3 Analysis

In Figure 2c, we investigate the impact of tuning γ in *MixedSP* on the dataset ALG-514. Recall that γ controls the fraction of the training time that the model uses solely explicit supervision. At first glance, it may appear that we should utilize the im-

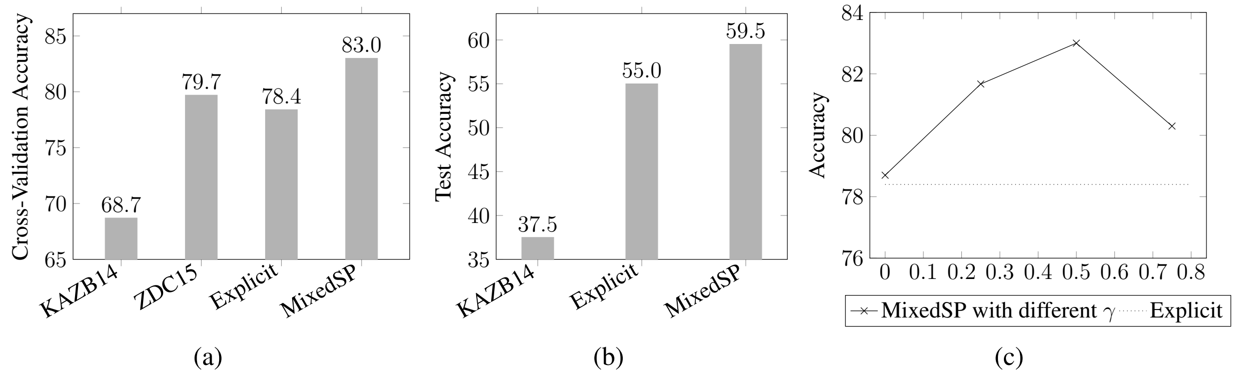

Figure 2: **(a)** Comparisons between our system to state-of-the-art systems on ALG-514. ZDC15 is the system proposed in (Zhou et al., 2015), and KAZB14 is the system proposed in (Kushman et al., 2014). **(b)** Comparisons between our system and other systems on DRAW-1K. Note that we are not able to run ZDC15 on DRAW-1K because it cannot handle some equation systems in the dataset. **(c)** Analysis of the impact of γ in MixedSP.

plicit supervision throughout training (set $\gamma = 0$). But setting γ to 0 hurts overall performance, suggesting in this setting that the algorithm uses a weak model to guide the exploration for using implicit supervision. On the other hand, by delaying exploration ($\gamma > 0.5$) for too long, the model could not fully utilize the implicit supervision. We observe similar trend on DRAW-1K as well. We found $\gamma = 0.5$ works well across the experiments.

We also analyze the impact of the parameter K, which controls the size of the candidate set Ω in *MixedSP*. Specifically, for DRAW-1K, when setting K to 5 and 10, the accuracy of *MixedSP* is at 59.5%. On setting K to 15, the accuracy of *MixedSP* improves to 61%. We suspect that enlarging K increases the chance to have good structures in the candidate set that can match the correct responses.

7 Conclusion

In this paper, we propose an algorithmic approach for training a word problem solver based on both explicit and implicit supervision signals. By extracting the question answer pairs from a Web-forum, we show that the algebra word problem solver can be improved significantly using our proposed technique, surpassing the current state-of-the-art.

Recent advances in deep learning techniques demonstrate that the error rate of machine learning models can decrease dramatically when large quantities of labeled data are presented (Krizhevsky et al., 2012). However, labeling natural language data has been shown to be expensive, and it has become one of the major bottleneck for advancing natural language understanding techniques (Clarke et al., 2010). We hope the proposed approach can shed light on how to leverage data on the web, and eventually improves other semantic parsing tasks such as knowledge base question answering and mapping natural instructions to actions.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. In *Proc. of TACL*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proc. of ICML*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proc. of EMNLP*.

Daniel G. Bobrow. 1964. A question-answering system for high school algebra word problems. In *Proceedings of the October 27-29, 1964, Fall Joint Computer Conference, Part I*.

K.-W. Chang, R. Samdani, and D. Roth. 2013. A constrained latent variable model for coreference resolution. In *Proc. of EMNLP*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. of ACL*.

Peter Clark and Oren Etzioni. 2016. My computer is an honor student-but how intelligent is it? Standardized tests as a measure of AI. *AI Magazine.*, 37(1).

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Proc. of CoNLL*.

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proc. of ICML*.

M. Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

R. Ge and R. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proc. of ACL*.

Javad Mohammad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proc. of EMNLP*.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? Large-scale dataset construction and evaluation. In *Proc. of ACL*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Proc. of TACL*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proc. of NIPS*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proc. of ACL*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*.

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

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artif. Intell. Rev.*, 29(2):93–122.

Allen Newell, John C Shaw, and Herbert A Simon. 1959. Report on a general problem-solving program. In *IFIP Congress*, pages 256–264.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proc. of EMNLP*.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proc. of EMNLP*.

J. Steinhardt and P. Liang. 2015. Learning with relaxed supervision. In *Proc. of NIPS*.

Sympy Development Team, 2016. *SymPy: Python library for symbolic mathematics*.

B. Taskar, C. Guestrin, and D. Koller. 2004. Max-margin markov networks. In *Proc. of NIPS*.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proc. of ICML*.

Shyam Upadhyay and Ming-Wei Chang. 2016. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *https://aka.ms/derivationpaper*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proc. of ACL*.

C. Yu and T. Joachims. 2009. Learning structural SVMs with latent variables. In *Proc. of ICML*.

J. M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic proramming. In *Proc. of AAAI*.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *EMNLP-CoNLL*.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proc. of EMNLP*.