ComSearch: Equation Searching with Combinatorial Mathematics for Solving Math Word Problems with Weak Supervision

Anonymous ACL submission

Abstract

Previous studies have introduced a weakly-supervised paradigm for solving math word problems requiring only the answer value annotation. While these methods search for correct value equation candidates as pseudo labels, they search among a narrow sub-space of the enormous equation space. To address this problem, we propose a novel search algorithm with combinatorial mathematics **ComSearch**, which can compress the search space by excluding mathematical equivalent equations. The compression allows the searching algorithm to enumerate all possible equations and obtain high-quality data. We investigate the noise in the pseudo labels that hold wrong mathematical logic , which we refer as the *false-matching* problem, and propose a ranking model to denoise the pseudo labels. Our approach holds a flexible framework to utilize two existing supervised math word problem solvers to train pseudo labels, and both achieve state-of-the-art performance in the weak supervision task.

1 Introduction

Solving math word problems (MWPs) is the task of extracting a mathematical solution from problems written in natural language. In Figure 1, we present an example of MWP. Based on a sequence-to-sequence (seq2seq) framework that takes in the text descriptions of the MWPs and predicts the answer equation (Wang et al., 2017), task specialized encoder and decoder architectures (Wang et al., 2018b, 2019; Xie and Sun, 2019; Liu et al., 2019; Guan et al., 2019; Zhang et al., 2020b,a; Shen and Jin, 2020), data augmentation and normalization (Wang et al., 2018a; Liu et al., 2020), pretrained models (Tan et al., 2021; Liang et al., 2021; Shen et al., 2021) and various other studies have been conducted on *full supervision* setting of the task. This setting requires equation expression annotation, which is expensive and time-consuming.

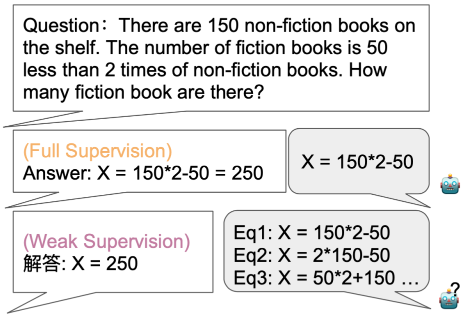

Figure 1: Example of MWP solving system under full supervision and weak supervision.

Recently Hong et al. (2021) and Chatterjee et al. (2021) addressed this problem and proposed the *weak supervision* setting, where only the answer value annotation is given for supervision. These methods first extract candidate equations that obtain the correct value and then use them as pseudo labels to train the MWP solving model. However, the solution space is enormous with the bruce-force searching used in these two studies, i.e., $O(n^{2n})$ with n variables. When the number of variables increases, it becomes computationally impossible to traverse all possible equations due to the high computational complexity. Hong et al. (2021) searches among neighbour equations of the wrong model prediction in the solution space via random walk. Chatterjee et al. (2021) trains a candidate equation extraction model using reinforcement learning (RL) to explore the solution space, where the reward is given by whether the equation obtains the correct value. Both these methods lack robustness and highly relies on initialization or beam searching.

We observe that although the search space is ample, many equations in the search space are equivalent. For example, in Figure 1, '$150 * 2 - 50$' and '$2 * 150 - 50$' are mathematically equivalent. Eliminating such equivalent expressions in the searching algorithm can compress the search space and lower the computational complexity. Roy and Roth (2015) proposed a model that decomposes the

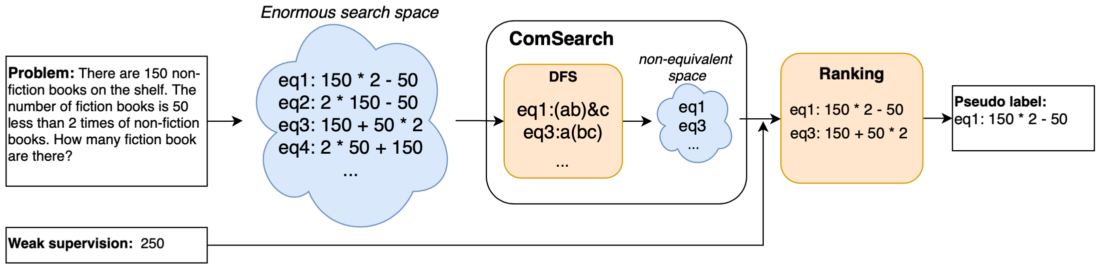

Figure 2: The model overview.

equation prediction problem to various classification problems, which eliminates some equivalence forms of the equation. However the compression is highly integrated with their model and cannot generalize to other models including the SOTA seq2seq based models. Moreover, it can only cover limited equivalence forms, leaving out various important forms such as Commutative law and Associative law. (Wang et al., 2018a) proposed a normalization method for supervised MWP systems that considers Commutative law. The method merges several equivalent expression to one expression, resulting in compression of the target equation space. However, their method requires bruce-force enumeration before compression, which remains to have high computational complexity. In both two studies, only limited equivalence forms are considered, that the equation space is still considerably large.

In this paper, we investigate theories in combinatorial mathematics and propose a new searching method that searches through only *non-equivalent equations* in the search space. Our method could be proven to have an approximate complexity of $O(n^n)$, allowing the algorithm to find all possible candidate equations with the given variables. We show that 77.5% percent of the examples have only one equation candidate and form high quality and reliable data.

We also observe a *false-matching* problem in the weakly supervised setting, where the candidate equation extraction algorithms reaches one or more candidates with the correct value, however their mathematical reasoning logic is wrong. For example, in Figure 1, '$150 * 2 - 50$' (Eq1) and '$50 * 2 + 150$' (Eq3) have the same value, however Eq3 holds a false mathematical reasoning logic and only Eq1 is correct. While previous methods (Hong et al., 2021; Chatterjee et al., 2021) take in all candidates with the correct value as annotated data, these annotations leads to false mathematical inference brings in noise to the training process. To address this problem, we build a ranking module to choose the best pseudo label for examples with multiple candidate equations. The module first searches for candidate equations via searching algorithms and generalization models, and then uses a classifier to choose the best candidate equation for the example. We investigate how the *false-matching* problem drags down the system's performance and propose two ranking models to alleviate this problem.

We conduct experiments on two strong MWP solvers, the results demonstrate the effectiveness and generalization ability of our method, achieving state-of-the-art (SOTA) results under the weakly supervised setting.

In summary, our contribution is three-fold:

- We propose ComSearch, which is a searching algorithm that enumerates non-equivalent equations without repeating to search effectively for candidate equations.
- We are the first to investigate the *false-matching* problem that brings noise to the pseudo training data. We propose a ranking module to reduce the noise and give detailed oracle analysis on the problem.
- We perform experiments on two MWP solvers with our ranking module, and achieve SOTA performance under weak supervision.

2 Methodology

We show the pipeline of our method in Figure 2. Our method consists of three modules: The Search with Combinatorial Mathematics (**ComSearch**) module that searches for candidate equations; the MWP model that is trained to predict equations given the natural language text and pseudo labels; the Ranking module that uses an explorer model to find candidate equations and select the best candidate equation with a scorer model.

Algorithm 1 *enum_skel(n)*

Require: $n \geq 1$

Intialize empty list skels
for $i \leq n$; $i = 1$; $i++$ **do**
    left_list = *unit_skel(i)*
    right_list = *enum_skels*$(n - i)$
    for *left* in *left_list* **do**
        for *right* in *right_list* **do**
            move the start index of *right* to i
            new_skels += *left* + *right*
        end for
    end for
    skels += *new_skels*
end for
return *skels*

2.1 ComSearch

Directly searching for non-equivalent equation expressions is difficult, because the searching method needs to consider Commutative law, Associative law and other equivalent forms. To enumerate all non-equivalent equations for four arithmetic operations, we transform the problem to finding skeleton structures that could be enumerated without repeat via deep-first search.

Definition We define the set of *non-equivalent equations* using four arithmetic operations as S_n. We sort the set to two categories, either $S^{\pm}$ where the outermost operators are $\pm$, such as $a/b - c + d$ and $a + (b * c - d)$, or $S^{\divideontimes}$ where the outermost operators are $\divideontimes$, such as $(a + b) * (c - d/e)$ and $b * (a - c)$. We call the former a *general addition equation* and the latter a *general multiplication equation*:

$$S_n^{\pm} = \{(x_1 \divideontimes (..)) \pm (x_i \divideontimes (..)) \pm ..x_n\} \quad (1)$$

$$S_n^{\divideontimes} = \{(x_1 \pm (..)) \divideontimes (x_i \pm (..)) \divideontimes ..x_n\} \quad (2)$$

These two sets are symmetrical. Consider elements in $S_n^{\pm}$, we can rewrite the equation to x. Thus we can form a mapping $g : x \to g(x)$ from an general addition equation x to an skeleton structure expression $g(x)$. :

$$\begin{aligned} x =& ((x_i \divideontimes (..)) + (x_j \divideontimes (..)) + ..) \\ & - ((x_k \divideontimes (..)) + (x_l \divideontimes (..)) + ..) \\ g(x) =& (x_i(..))(x_j(..))..\&(x_k(..))(x_l(..)).. \end{aligned}$$

The order of x_i within the same layer of brackets is ignored in $g(x)$, that it can deal with the equivalence caused by Commutative law and Associative law. The addition and substraction terms are split by $\&$, that it can deal with equivalence cause by removing brackets. $g(x)$ is a bijection, so the enumeration problem transforms to finding such skeletons:

$$\begin{aligned} n = 1 :& a \\ g^{-1} :& a \\ n = 2 :& ab, a\&b, b\&a \\ g^{-1} :& a + b, a - b, b - a \\ n = 3 :& abc, a\&(b\&c), (ab)\&c, ... \\ g^{-1} :& a + b + c, a - (b/c), (a * b) - c, ... \\ & ... \end{aligned}$$

The enumeration problem of these structures is an expansion of solving Schroeder's fourth problem (Schröder, 1870), which calculates the number of labeled series-reduced rooted trees with n leaves. We use a deep-first search algorithm shown in Algorithm 1 to enumerate these skeletons. It considers the position of the first bracket and then recursively finds all possible skeletons of sub-sequences of the variable sequence $\mathcal{X} = \{x_k\}_{k=1}^{i}$ (Wang, 2021).

While considering such skeletons could enumerate all unique expressions, equations have at least one element on the left of $\&$ in our target domain and do not start with $-$ or $\div$. We further extend the algorithm to consider these cases. To be noticed, because there is at least one $+$ or $*$ operator for each equation, the left side of $\&$ must not be empty while the right part has no restrictions. Thus we define the *unit_skel(i)* equation to return possible skeletons with only one or none $\&$ and no brackets. This constraint is equivalent to finding non-empty subsets and its complement of the variable sequence $\mathcal{X}$. We can use Algorithm 1 to perform the enumeration of such skeletons, except for defining two different *unit_skel(i)* to support the enumeration of subtraction and division operation. The enumeration algorithm of non-empty subsets is trivial and omitted here.

$$unit\_skel_{div}(i) = \{(A\&\overline{A}) | A \subseteq \mathcal{X}; A \neq \emptyset\} \quad (3)$$

$$\begin{aligned} & unit\_skel_{sub}(i) = \\ & \quad \{((a(A - a))\&\overline{A - a}) | A \subseteq \mathcal{X}; a \in A\} \end{aligned} \quad (4)$$

We transform the skeletons back to equations to obtain all non-equivalent equations S_n. Such enumeration considers absolute values and omits pairs of solutions that are opposite to each other. To search effectively, for the equations that contain substraction, we add their opposite equation to the searching space. Given the compressed search space, we substitute the values for variables in the equation templates and use the equations which value matches with the answer number as candidate equations. If no equations could be extracted by using all numbers, we continue to consider: 1.omitting one number, 2.adding constant number 1 and pi and 3.using one number twice. If the algorithm extracts candidates at any stage, the further stages are not considered since it would introduce repeating equations, e.g. $1 * (a + b)$ is a duplication of $a + b$.

2.2 MWP Solving Models

2.2.1 Goal-driven Tree-structured Solver

We follow Hong et al. (2021) and Chatterjee et al. (2021) and use Goal-driven tree-structured MWP solver (GTS) (Xie and Sun, 2019) as the MWP model. GTS is a seq2seq model with the attention mechanism that uses a bidirectional long short term memory network (BiLSTM) as the encoder and LSTM as the decoder. GTS also uses a recursive neural network to encode subtrees based on its children nodes representations with the gate mechanism. With the subtree representations, this model can well use the information of the generated tokens to predict a new token.

Formally, the model takes a sequence of tokens $\{x_i\}_{i=0}^n$ as the input, the encoder is a bidirectional LSTM with hidden states h_i^{enc} and the decoder adopts a unidirectional LSTM to generate the output in an autoregressive manner. Given decoder hidden states $\{h_t^{dec}\}$, GTS also considers subtree representations which can provide more information for the decoding process. A recursive neural network is used to encode subtrees of the equation in a bottom-up manner. The subtree representation $e_t^{subtree}$ at timestep t is calculated based on its children nodes representations with the gate mechanism. With the subtree representations, this model can also well use the information of the generated tokens to predict a new token. The representations s_t and $e_t^{subtree}$ are finally fed to a Multi-layer Perceptron (MLP) layer to generate the output token y_t.

$$\begin{aligned} s_t &= \sum_{i=1}^{n} \alpha_t^i \cdot h_i^{enc} \\ &= \sum_{i=1}^{n} \frac{exp(h_i^{enc} \cdot h_t^{dec})}{\sum_{j=1}^{n} exp(h_j^{enc} \cdot h_t^{dec})} \cdot h_i^{enc} \end{aligned} \quad (5)$$

2.2.2 Graph-to-Tree Solver

Following Chatterjee et al. (2021), we conduct experiments on Graph-to-Tree (G2T) Solver (Zhang et al., 2020b) . G2T is a direct extension of GTS, which consists of a graph-based encoder capturing the relationships and order information among the quantities.

Formally, given the encoder hidden states in Equation **??**, the model uses GCNs built on the Quantity Cell Graph and the Quantity Comparison Graph to calculate the graph embedding h_i^g of each token i. The Quantity Cell Graph aims to associate informative descriptive words to quantity so as to enrich the quantity's representation, while the Quantity Comparison Graph aims to retain the numerical qualities of the quantity and leverage heuristics to improve representations of the relationships among quantities. Given the Graph G_k:

$$\{h_i^{g_k}\}_{i=0}^n = GCN(G_k, \{h_i^{enc}\}_{i=0}^n) \quad (6)$$

The final token representation h_i^g is the concatenation of the two graphs. The decoder part is similar to GTS.

2.3 Ranking

While ComSearch enumerates equations that are non-equivalent without repeat, some variable sets can coincidentally form multiple equations with the same correct value, as we show in Figure 2. The equations $150 * 2 - 50$ and $150 + 50 * 2$ are non-equivalent, their values are equal, while only $150 * 2 - 50$ is the correct solution. We refer this problem as *false-matching*, which is an important issue that has been overlooked by previous studies. Previous work do not perform any processing on these *false-matching* examples, which brings in noise to the training data.

To process these data that have multiple candidate equations, we propose two ranking methods to choose the best candidate equation. The methods is consist of two component, the *explorer* that searches for candidate equations via searching algorithms and generalization models, and the *scorer* that classifies which candidate equation is the best annotation for the example.

In the first method which we call the *Check Ranker*, we use ComSearch as the *explorer*. We assume that the pseudo data with only one equation matching the answer is reliable, and leverage this data to train the MWP model J, which is used as the scorer. For each candidate text and equation

Model	Term	#	Prop(%)
-	All Data	23,162	-
Ours	Too Long	233	1.0
	Power Operator	51	0.2
	Single	17,959	77.5
	Multiple	3,931	17.0
	Data	21,890	**94.5**
WARM	Data (w/o beam)	-	14.5
	Data (w/ beam)	-	80.1

Table 1: Statistics of ComSearch Results.

pair x and its corresponding ComSearch candidate equations $\{y_{eq}\}^{search}$, we check the score of each candidate equation in the set to obtain the candidate equation with the best score. The scorer model predicts the score of the equation at each time step t and sum the logarithm of the scores together.

$$s_{eq} = \sum_{i=0}^{k} log(J(x, y_{eq})) \tag{7}$$

We use the candidate equation that has the highest score as the pseudo label of this example and add it to the training data.

In the second method which we call the *Beam Ranker*, we use both the MWP model and ComSearch as the *explorer*. We observe a high precision on the predictions of the MWP model J when the answer is correct, that these prediction can also form candidate equations. We perform beam search with J and add the predictions that hold a correct answer $\{y_{eq}\}^{beam}$ to the candidate equation set. We build an simple beam-score based judge model for this approach. If $\{y_{eq}\}^{beam}$ is not an empty set, the highest beam score prediction is considered as the best candidate equation. If $\{y_{eq}\}^{beam}$ is empty, we consider the ComSearch results that has the highest beam score. Here we use the same model J and score s_{eq} to calculate the beam score.

3 Experiments

3.1 Dataset and Baselines

We evaluate our proposed method on the Math23K dataset. It contains 23,161 math word problems annotated with solution expressions and answers. We only use the problems and final answers. We evaluate our method on the train-test split setting of Wang et al. (2018a). All the results are evaluated by the three-run average.

We compare our weakly-supervised models' math word problem solving accuracy with two baselines methods.

Chatterjee et al. (2021) proposed **WARM** that uses RL to train an equation candidate generation model with the reward of whether the value of the equation is correct. Since the reward signal is sparse due to the enormous search space, the top1 accuracy of the candidate generation model is limited, it uses beam search to further search candidates.

Hong et al. (2021) proposed **LBF**, a learning-by-fix algorithm that searches in neighbour space of the predicted wrong answer by random walk and tries to find a fix equation that holds the correct value as the candidate equation. *memory* saves the candidates of each epoch as training data.

3.2 Analysis on ComSearch

3.2.1 Search Statistics

We give statistics of ComSearch in Table 1. Among the 23,162 examples, 233 have more than 6 variables that we filter them out, and 51 use the power operation that our method is not applicable. 94.5% of the examples find at least one equation that can match the answer value, significantly higher than WARM, which covers only 80.1% of the examples. LBF dynamically searches for candidate equations, and this measurement is not applicable. 17,959 examples match with only one equation, and 3,931 examples match with two or more equations that need the ranking module to choose the pseudo label further. We show the distribution of these examples in the appendix.

3.2.2 Eliminating Equivalent Equations in Search Space

We show the empirical compression of the search space with ComSearch in Table 2. As we can see, the compression ratio of ComSearch increases as the variable number grows, up to more than 100 times when the number of variables reaches 6. We also show the results of considering removing brackets, where $-/\div$ can not be the children node of $+/*$, which is the compression considered in Roy and Roth (2015); and Commutative Law, which is the compression considered in Wang et al. (2018a). We show that although the two methods can compress the search space to some extent, but there is a large gap between their compression efficiency and ours, up to more than 20 times when the number of variables reaches 6.

The size of the Bruce-Force search space could be directly calculated, which is $n! * (n-1)! * 4^{n-1}$.

#Variable	Bruce-Force	Removing Brackets	Commutative	ComSearch	Ratio
1	1	1	1	1	1
2	8	8	6	6	1.3
3	192	144	108	68	2.8
4	9,216	5,184	3,816	1,170	7.9
5	737,280	311,040	224,640	27,142	27.2
6	88,473,600	27,993,600	19,841,760	793,002	111.6

Table 2: Empirical Results of Search Space Size.

Model	**Valid(%)**	**Test(%)**
GTS based		
WARM	-	12.8
+*beam*	-	54.3
LBF	57.2($\pm$0.5)	55.4($\pm$0.5)
+*memory*	56.6($\pm$6.9)	55.1($\pm$6.2)
Ours	**61.0**($\pm$0.3)	**60.0**($\pm$0.3)
GTS	-	75.6
G2T based		
WARM	-	13.5
+*beam*	-	56.0
Ours	**61.7**($\pm$1.1)	**60.5**($\pm$0.6)
G2T	-	77.4

Table 3: Results on Math23K. $\pm$ denotes the variance of 3 runs for valid/test

Model	**Valid(%)**	**Test(%)**
Single Equation	58.9	57.5
Random Sample	57.3	56.3
Check Ranker	60.1	59.2
Beam Ranker	61.0	60.0

Table 4: Results of Ablation Study for Ranking. 'Random Sample' denotes removing the ranking module and randomly sampling an equation for the examples that match with two or more equations.

If we consider the exponential generating function of $card(S_n)$, based on Smooth Implicit-function Schema, we can have an approximation of S_n: $card(S_n) \sim C * n^{n-1}$, which shows our searching method compresses the search space more than exponential level. We give a proof in the appendix.

3.3 Main Results and Ablation Study

We show our experimental results in Table 3. We reproduced the results of LBF with their official code and found that LBF+memory lacks robustness. As we can see in the table, the performance of LBF has high variance on both validation and test set. For a fair comparison, we additionally ran 5-fold cross validation setting according to (Hong et al., 2021) for our model and LBF+memory with the GTS model. The results show that LBF + memory achieves cross-validation score of 56.3% with variance of $\pm$6.2, while our model achieves cross-validation score of 59.7 with variance of $\pm$1.0, which performs similar to the train-test setting. We observe that its performance highly relies on the initialization of the model. When fewer candidates are extracted at early-stage training, the performance drops drastically since LBF relies on random walks in an enormous search space. Our method achieves state-of-the-art performance and outperforms other baselines up to 3.8% and 2.7% on train-test and cross-validation settings. Our method is also more robust with minor variance.

We perform an ablation study with the GTS based train-test setting in Table 4. *Single Equation* denotes using the 17,959 examples that only match with one equation, the model achieves 57.5% performance, which is slightly lower than using all data and the ranking module, out-performing other baseline models. This shows that the examples with only one matching could be considered highly reliable and achieve comparable performance with a smaller training data size. We observe a performance drop of at least 2.9% point without the ranking module, showing that our ranking module improves the performance. We can see that there is a performance gap of 0.9% between the two rankers, demonstrating the importance of considering candidate equations from the model prediction.

3.4 Analysis

We conduct analysis on GTS train-test setting since the model achieve similar performance compared with G2T and the run time is less.

3.4.1 Study on Number of Variables

In Table 5, we show the comparison of model performance on examples of a different number of variables. For the examples with 1 or 2 variables, LBF has a slight performance advantage. While

#Var	LBF(%)	ComSearch(%)	Prop(%)
1	**75.0**	50.0	1.6
2	**75.2**	73.4	33.1
3	56.2	**62.9**	48.5
4	4.8	**25.8**	12.4
5	3.2	**16.1**	3.1
6	0	**28.6**	0.7
7	0	**25.0**	0.4

Table 5: Results of different number of variables.

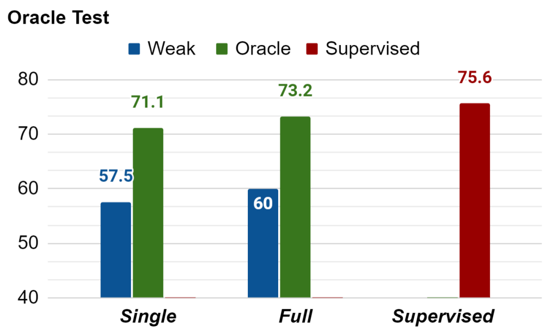

Figure 3: Results of Oracle Test with gold labels.

the variable number grows, our method achieves better performance on examples with more variables and larger search space, which demonstrates the efficiency of ComSearch. Eliminating equivalent equations allows our method to consider the larger search space, while LBF limits to a small neighbour space of the model prediction. When the variable number is small, the in-place random walk of LBF can possibly cover the correct equations such as not using all numbers. When the variable number grows larger, as we show in Table 2, the gap between the efficiency of our searching method and LBF expands, our method can consider more equations candidates and achieve better performance.

3.4.2 Oracle Test

While our searching method covers 94.5% of the training data as shown in Table 1, there is still a significant performance gap between the weakly supervised performance and fully supervised performance. As we stated in Section 2.3, we observe that the *false-matching* problem could potentially draw down the performance, which is verified by the effectiveness of the ranking module.

To further analyze our two modules, we perform two oracle tests for the weakly supervised system. In Figure 3, using the same data examples, we replace the weakly supervised annotations with the supervised gold labels and train the MWP solver. We can see that there is a performance gap of around 10% using the same data examples as training data, which indicates that the weakly supervised annotations contains noise. Since all candidate equation annotations have the correct answer, the *false-matching* problem is the reason that this noise exists. This can show that the *false-matching* problem is the key issue in weakly supervised setting that causes the performance gap compared to supervised setting.

Model	Equation Acc(%)
Single	65.5
Multiple	2.7
Full	23.0
Ranker 1(Multiple)	45.6
Ranker 2(Multiple)	57.4
Ranker 2(Full)	63.4

Table 6: Equation accuracy of different methods.

In Table 6, we show the results of equation accuracy of the training data. We check whether the pseudo annotations that our system obtains is equivalent to the gold labels, and the accuracy is calculated based on candidate equation level. We can see that even in the examples that can only extract one candidate equation, the error rate is still relatively high. We show examples in the case study section to explain this problem. The examples that extract more than one candidate has an equation accuracy rate as low as 2.7%, which makes our ranking system essential. Benefited from the ranking system, the multiple candidate data can also achieve a higher equation accuracy rate. The second ranker performs better than the first considering beam search results.

3.4.3 Case Study

We conduct case study for ComSearch on three examples to further discuss the strengths and limitations of the method in Table 7.

The first example extracts only one candidate equation, although the written expression is different from the gold label, the two equations are equivalent and the candidate is true-matching. The second example extracts only one candidate equation, the *false-matching* candidate coincidentally equals to the correct answer with this set of number, however the candidate expression and gold label expression are not equivalent. The algorithm reaches a candidate at the stage of using all numbers and does not further search for candidates that use the constant number 1. The third example extracts

Text	Candidates	Gold	Ans
Some children are planting trees along a road every 2 meters. They plant trees on both ends of the road. At last they planted 11 trees. How long is the road?	2*(11-1)	(11-1)*2	20
A library has 30 books. On the first day, $\frac{1}{5}$ of the books were borrowed out. On the second day, 5 books were returned. How many book are there in the library now?	30 - $\frac{1}{5}$ * 5	30*(1-($\frac{1}{5}$)) + 5	29
Peter and a few people are standing in a line, one person every 2 meters. Peter found that there are 4 people before him and 5 people after him. How long is this queue?	4*5-2 , (4+5)*2	4*2 + 5*2	18

Table 7: Case study of ComSearch. The dark green color denotes that the candidate is true-matching and the light red color denotes that the candidate is false-matching.

two candidate equations, while only $(4+5)*2$ holds the correct mathematical knowledge. The two candidates appear at the same searching stage and such *false-matching* cannot be avoid by our current searching method, where we need the ranker to help filter out the *false-matching* noise. For this example, the two rankers both select the correct label.

4 Related Work

Early approaches on math word problems mainly depend on hand-craft rules and templates (Bobrow, 1964; Charniak, 1969). Later studies either use parsing methods, which relies on semantic parsing (Roy and Roth, 2018; Shi et al., 2015; Zou and Lu, 2019), or try to obtain an equation template (Kushman et al., 2014; Roy and Roth, 2015; Koncel-Kedziorski et al., 2015; Roy and Roth, 2017). Recent studies focus on using deep learning models to predict the equation template for full supervision setting.

For weakly supervised setting, Hong et al. (2021) and Chatterjee et al. (2021) suffers from two major drawbacks. First they apply equation candidate searching on an enormous searching space, while our method can effectively extract high quality candidate equations. Hong et al. (2021) results in low robustness and low performance on examples with more variables. Chatterjee et al. (2021) results in low coverage of examples that can extract candidate equation. Second they use all candidate equations for training and neglect the *false-matching* problem, which is the key issue that drags down the model performance in weakly supervised setting, while our ranking module addresses this issue and further boosts the performance.

For eliminating equivalent expressions, Roy and Roth (2015) cannot generalize to SOTA models; Wang et al. (2018a) performs compression instead of enumeration without repeat of the target equation space, which remains to have high computational complexity. Both methods only consider a part of the equivalence forms which leads to limited compression efficiency.

5 Conclusion and Future Work

This paper proposes ComSearch, a searching method based on Combinatorial Mathematics, to extract candidate equations for Solving Math Word Problems under weak supervision. ComSearch compresses the enormous search space of equations beyond the exponential level, allowing the algorithm to enumerate all possible non-equivalent equations to search for candidate equations. We investigate the *false-matching* problem, which is the key issue that drags down performance, and propose a ranking model to reduce noise. Our experiments show that our method obtains high-quality pseudo data for training, achieves state-of-the-art performance under weak supervision settings, outperforming strong baselines, especially for the examples with more variables.

As we observe from experiments, the performance gap between the most reliable weak data and oracle data is still 10% and the noise rate in the pseudo data is still relatively high. Meanwhile our ranking module only denoises multiple candidate equations examples. For future work, we would consider applying more advanced learning from noise algorithms and denoise more training data.

A Proof for Search Space Approximation

Because there is at least one $+$ or $*$ operator for each equation (i.e. $-a-b-c$ is illegal), the target S_n is not symmetric and is hard to directly approximate. We need two assisting targets to form the approximate. This proof majorly relies on Flajolet and Sedgewick (2009).

We first consider target U that considers only $+$, $*$ and $\div$ three operators. We sort it into two categories: U^+ that the outermost operator is $+$ and U^* that the outermost operator is $\circledast$. Equations such as $\frac{1}{a} * \frac{1}{b-c}$ are still considered illegal.

Z corresponds to a single variable equation. We can have the construction of U:

$$U^+ = Z + SET_{\geq}(U^*) \tag{8}$$
$$U^* = Z + (2^2-1) * SET_{=2}(U^+) \tag{9}$$
$$+ (2^3-1) * SET_{=3}(U^+)... \tag{10}$$

We apply symbolic method to obtain the EGF of the constructions:

$$U^+(z) = z + \sum_{k\geq 2} \frac{1}{k!}[U^*(z)]^k \tag{11}$$
$$= z + [e^{U^*(z)} - 1 - U^*(z)] \tag{12}$$
$$U^*(z) = z + \sum_{k\geq 2} \frac{2^k-1}{k!}[U^+(z)]^k \tag{13}$$
$$= z + e^{2U^+(z)} - e^{U^+(z)} - U^+(z) \tag{14}$$

Meanwhile we have:

$$U(z) = U^+(z) + U^*(z) - z \tag{15}$$

Next we consider target T that $-a-b-c$ is considered legal. Similarly we define $T^\pm$ and T^*. We consider the construction:

$$T^\pm = 2Z + SET_{\geq}(T^*) \tag{16}$$
$$T^* = 2Z + 2[(2^2-1) * SET_{=2}(T^\pm/2) \tag{17}$$
$$+ (2^3-1) * SET_{=3}(T^\pm/2)...] \tag{18}$$

With symbolic method we have:

$$T^\pm(z) = 2z + \sum_{k\geq 2} \frac{1}{k!}[U^*(z)]^k \tag{19}$$
$$= 2z + [e^{T^*(z)} - 1 - T^*(z)] \tag{20}$$
$$T^*(z) = 2z + 2\sum_{k\geq 2} \frac{2^k-1}{k!}[T^\pm(z)/2]^k \tag{21}$$
$$= 2z + 2e^{T^\pm(z)} - 2e^{T^\pm(z)/2} - T^\pm(z) \tag{22}$$

The illegal equations such as $-a-b-c$ in T equals to the counts of $a+b+c$, which is actually U. So we have:

$$S(z) = T(z) - U(z) \tag{23}$$

We now have the EGF of S_n.

With Smooth implicit-function schema and Stirling approximiation function we have, for an EGF $y(z) = \sum_{n\geq 0} y_n z^n$, Let $G(z,w) = \sum_{m,n\geq 0} g_{m,n} z^m w^n$, thus $y(z) = G(z, y(z))$:

$$n! * [z^n]y(z) \sim \frac{c * n!}{\sqrt{2\pi n^3}} * r^{-n+1/2} \tag{24}$$
$$\sim \frac{c\sqrt{2\pi n r}}{\sqrt{2\pi n^3}} (\frac{1}{r})^n (\frac{n}{e})^n \tag{25}$$
$$= \frac{c\sqrt{r}}{n} (\frac{n}{re})^n \tag{26}$$

while r:

$$G(r,s) = s \tag{27}$$
$$\frac{\partial G(r,s)}{\partial w} = 1 \tag{28}$$

and c:

$$c = \sqrt{\frac{\partial G(r,s)/\partial z}{\partial^2 G(r,s)/\partial w^2}} \tag{29}$$

We still need the two assisting targets to perform the approximation. We have:

$$U^+(z) = e^{z + e^{2U^+(z)} - e^{U^+(z)} - U^+(z)} \tag{30}$$
$$- e^{2U^+(z)} + e^{U^+(z)} + U^+(z) - 1 \tag{31}$$

Let $G(z,w) = z + e^{2w} - e^w - ln(1 + e^{2w} - e^w)$, considering 27 and 29, r, s and c would be constant numbers.

So we have:

$$n![z^n]U^+(z) \sim \frac{c_1\sqrt{r_1}}{n}(\frac{n}{r_1 e})^n \tag{32}$$

Similarly we can approximate U^*, $T^\pm$ and T^*:

$$n![z^n]U^*(z) \sim \frac{c_2\sqrt{r_1}}{n}(\frac{n}{r_2 e})^n \tag{33}$$
$$n![z^n]T^\pm(z) \sim \frac{c_3\sqrt{r_2}}{n}(\frac{n}{r_3 e})^n \tag{34}$$
$$n![z^n]T^*(z) \sim \frac{c_4\sqrt{r_2}}{n}(\frac{n}{r_4 e})^n \tag{35}$$

So we have:

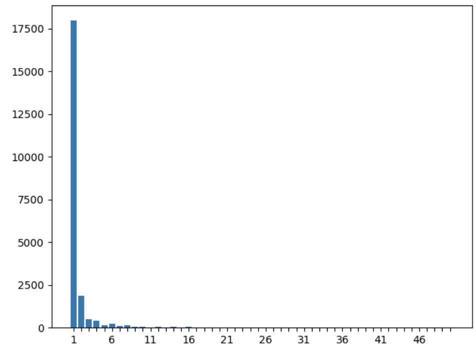

Figure 4: Distribution of Candidate Equation Number.

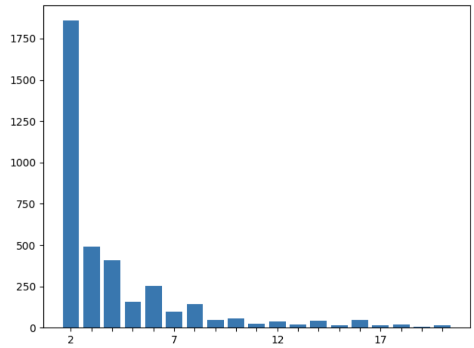

Figure 5: Distribution of Candidate Equation Number.

$$u_n = n![z^n]U(z) \sim \frac{(c_1 + c_2)\sqrt{r_1}}{n}\left(\frac{n}{r_1 e}\right)^n \quad (36)$$

$$t_n = n![z^n]T(z) \sim \frac{(c_3 + c_4)\sqrt{r_2}}{n}\left(\frac{n}{r_2 e}\right)^n \quad (37)$$

Since $S(z) = T(z) - U(z)$, the subtraction of u_n and t_n would be our approximation. However we observe that $r_1 \gg r_3$, that u_n can be ignored. So we have:

$$s_n = n![z^n]S(z) \sim \frac{(c_3 + c_4)\sqrt{r_2}}{n}\left(\frac{n}{r_2 e}\right)^n \quad (38)$$

Q.E.D.

B Distribution of Candidate Equations

The largest candidate equation number of one example is 3914. We show the distribution of candidate equations in Figure 4 and 5. The x axis represent the the number of candidate, while the y axis represents the number of examples that have x candidate equations. We can see from Figure 4, which includes examples that have 1 to 50 candidates, it is a long tail distribution that most examples only have a few candidate equations. From Figure 5, where we zoom in and focus on examples that have 2 to 20 candidates, we can see that there are a lot of examples that have more than 2 candidate equations, and the ranking module is essential.

C Experimental Details

We run our experiments on single card GTX3090Ti, each run takes around 2-3 hours for all models. We did not perform extra hyperparameter searching and use the same hyperparameters as the public release of the two models, except for epoch number which is decided by the validation set. The code is conducted based on Pytorch.

References

Daniel G. Bobrow. 1964. Natural language input for a computer problem solving system. Technical report, Cambridge, MA, USA.

L. Carlitz and J. Riordan. 1956. The number of labeled two-terminal series-parallel networks. *Duke Mathematical Journal*, 23(3):435 – 445.

Eugene Charniak. 1969. Computer solution of calculus word problems. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, IJCAI'69, pages 303–316, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Oishik Chatterjee, Aashish Waikar, Vishwajeet Kumar, Ganesh Ramakrishnan, and Kavi Arya. 2021. A weakly supervised model for solving math word problems.

Philippe Flajolet and Robert Sedgewick. 2009. *Analytic Combinatorics*. Cambridge University Press.

Wenyv Guan, Qianying Liu, Guangzhi Han, Bin Wang, and Sujian Li. 2019. An improved coarse-to-fine method for solving generation tasks. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 178–185, Sydney, Australia. Australasian Language Technology Association.

Yining Hong, Qing Li, Daniel Ciao, Siyuan Huang, and Song-Chun Zhu. 2021. Learning by fixing: Solving math word problems with weak supervision. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):4959–4967.

W. Knödel. 1951. Über zerfällungen. *Monatshefte für Mathematik*, 55:20–27.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

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

52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281.

Yihuai Lan, Lei Wang, Qiyuan Zhang, Yunshi Lan, Bing Tian Dai, Yan Wang, Dongxiang Zhang, and Ee-Peng Lim. 2021. Mwptoolkit: An open-source framework for deep learning-based math word problem solvers.

Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. 2019. Modeling intrarelation in math word problems with different functional multi-head attentions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6162–6167.

Zhenwen Liang, Jipeng Zhang, Jie Shao, and Xiangliang Zhang. 2021. Mwp-bert: A strong baseline for math word problems.

Qianying Liu, Wenyu Guan, Sujian Li, Fei Cheng, Daisuke Kawahara, and Sadao Kurohashi. 2020. Reverse operation based data augmentation for solving math word problems. *arXiv preprint arXiv:2010.01556*.

Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019. Tree-structured decoding for solving math word problems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2370–2379, Hong Kong, China. Association for Computational Linguistics.

John Riordan and Claude E Shannon. 1942. The number of two-terminal series-parallel networks. *Journal of Mathematics and Physics*, 21(1-4):83–93.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Subhro Roy and Dan Roth. 2018. Mapping to declarative knowledge for word problem solving. *Transactions of the Association of Computational Linguistics*, 6:159–172.

Ernst Schröder. 1870. Vier combinatorische probleme. *Zeitschrift für Mathematik und Physik*, 15.

Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. 2021. Generate & rank: A multi-task framework for math word problems. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2269–2279.

Yibin Shen and Cheqing Jin. 2020. Solving math word problems with multi-encoders and multi-decoders. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2924–2934, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1132–1142.

Minghuan Tan, Lei Wang, Lingxiao Jiang, and Jing Jiang. 2021. Investigating math word problems using pretrained multilingual language models.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018a. Translating a math word problem to a expression tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1064–1069.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018b. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7144–7151.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.

Yun Wang. 2021. *The Math You Never Thought Of*. Posts & Telecom Press Co., Ltd., Beijing.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 5299–5305.

Dongxiang Zhang, Lei Wang, Luming Zhang, Bing Tian Dai, and Heng Tao Shen. 2019. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE transactions on pattern analysis and machine intelligence*, 42(9):2287–2305.

Jipeng Zhang, Roy Ka-Wei Lee, Ee-Peng Lim, Wei Qin, Lei Wang, Jie Shao, and Qianru Sun. 2020a. Teacher-student networks with multiple decoders for solving math word problem. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial

Intelligence, IJCAI-20, pages 4011–4017. International Joint Conferences on Artificial Intelligence Organization. Main track.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020b. Graph-to-tree learning for solving math word problems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3928–3937.

Yanyan Zou and Wei Lu. 2019. Text2math: End-to-end parsing text into math expressions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5330–5340.