

Article

Syntax–Semantics–Numeracy Fusion for Improving Math Word Problem Representation and Solving

Zihan Feng, Hao Ming *  and Xinguo Yu

National Engineering Research Center for E-Learning, Central China Normal University, Wuhan 430079, China; zihanfeng@mails.ccnu.edu.cn (Z.F.); xgyu@ccnu.edu.cn (X.Y.)

* Correspondence: hming@ccnu.edu.cn

Abstract

Most pre-trained language representation models are designed to encode contextualized semantic information for general language processing tasks. However, they are insufficient for math word problem (MWP) solving, which requires not only linguistic syntax and semantic understanding but also numerical reasoning. In this work, we introduce SSN4Solver, a deep neural solver that improves MWP-solving performance by symmetrically fusing syntax, semantics, and numeracy representations within its contextual encoder. Our approach jointly captures syntactic structures from dependency trees, semantic features from part-of-speech tags, and the attributes and relations of numerical entities. By treating these heterogeneous information sources in a balanced and aligned manner, SSN4Solver constructs a rich, multi-faceted representation for MWP solving without introducing substantial computational overhead, empowering human–computer interaction (HCI) applications such as adaptive educational interfaces and intelligent tutoring systems. Extensive experiments demonstrate that SSN4Solver outperforms existing baseline models. In addition, a visualization scheme is designed to elucidate how the three types of representations contribute to the solving process. SSN4Solver thus offers a scalable solution, contributing to the development of HCI systems that are both intelligent and mathematically effective.

Keywords: math word problem; deep neural solver; representation model; numeracy learning; intelligent tutoring systems

1. Introduction

Math word problem (MWP) solving has long been recognized as a fundamental research domain due to its significant theoretical implications and practical applications. Beyond being a canonical benchmark for mathematical reasoning tasks, MWP solving also serves as a core computational capability in human–computer interaction (HCI) for education, e.g., intelligent tutoring systems, interactive practice platforms, and adaptive learning interfaces. The effectiveness of such systems depends on their ability to mirror human cognitive processes by creating the symmetry between user mental models and machine reasoning.

The deep learning-based solving paradigm [1,2] and the relation-flow solving paradigm [3,4] have emerged as two predominant research directions in developing MWP solvers. The deep learning direction has had notable contributions spanning from early sequence-to-sequence (seq2seq) models to representation-enhanced models based on Pre-training Language Models (PLMs) [5–7]; the relation-flow direction has had contributions advanced from novel methodological frameworks to theoretical algorithmic



Academic Editor: Carlo Cattani

Received: 20 January 2026

Revised: 18 February 2026

Accepted: 19 February 2026

Published: 2 March 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

foundations [8,9]. The development of deep neural solvers has become one of the primary approaches in MWP research, driven by the remarkable success of deep neural networks across various domains and their demonstrated advantages in accuracy, computational efficiency, and architectural elegance. The pioneering work by Wang et al. introduced DeepNS [1], establishing the foundational encoder–decoder framework for this task. Subsequent research has focused on enhancing these architectures through various innovations, including novel network structures [10], alternative learning methodologies [11], and diverse decoding objectives [5]. The emergence of large language models (LLMs) has further expanded this landscape, leading to the development of multiple PLM-based solvers for MWP [12,13]. However, researchers have recognized that LLMs, being primarily designed for linguistic tasks, exhibit limitations in capturing sophisticated mathematical representations [6,7]. This insight has prompted the creation of representation-enhanced PLM-based solvers.

Current deep neural solvers often suffer from informational asymmetry. While PLMs excel at linguistic representations, they often exhibit a bias towards textual patterns while neglecting the rigorous logic of numerical attributes. These limitations will also affect the effectiveness of the interactive mathematics intelligence tutoring system based on these representation models. This lack of balance creates a cognitive dissonance in HCI scenarios. As Figure 1 shows, traditional PLMs may have shortcomings in utilizing representations in solving math word problems, whereas several recent relation-flow algorithms have achieved state-of-the-art performance by intensively using diverse representations [9,14,15]. This shortcoming lies in that minor variations in keywords within an MWP can lead to completely different solution templates. However, the presence of bias in the training corpus can cause the model to learn flawed contextual dependencies, rendering it insensitive to certain key terms. Taking the BERT model as an example, we found a significant token prediction bias; the model is 30 times more likely to predict “more” than “fewer”. Such bias does not only degrade accuracy; it also amplifies informational asymmetry in HCI contexts, where users naturally expect consistent feedback under semantically equivalent formulations.

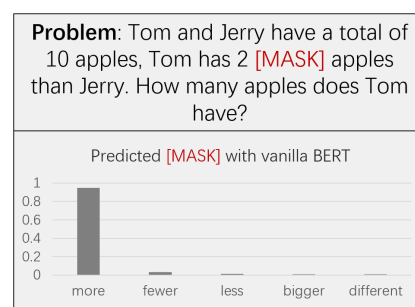


Figure 1. Demonstrating vanilla BERT’s bias against infrequent keywords by an example.

To work against these challenges, this paper introduces **SSN4Solver**, a novel deep neural network for MWP solving, which incorporates the diverse representations inspired by the relation-flow algorithms. Our model integrates three complementary types of representations, including linguistic syntactic structures (dependency trees), grammatical semantic part-of-speech (POS) features, and numerical reasoning, into a unified representational space to optimize embedding quality. By systematically infusing these triple representations with large language models within a cohesive training framework, SSN4Solver enhances the reasoning capabilities of existing solvers. Extensive experimental evaluations on standard MWP benchmarks demonstrate the effectiveness of our model in capturing multi-dimensional problem representations, which facilitates more accurate and robust solutions compared to strong baseline deep neural solvers. Although SSN4Solver is proposed

as an MWP solver, its architecture and training method can serve as technical building blocks for HCI systems in education.

The principal contributions of this work are as follows:

- An enhanced MWP solver is proposed that leverages BERT and RoBERTa-based encoders as the backbone. The model strengthens text encoding through multi-dimensional feature enhancement—covering syntax, semantics, and numeracy—to mitigate informational asymmetry and advances the capabilities of human–computer interaction applications.
- We introduce a syntax-regularized contrastive objective that builds positive/negative pairs using TF-IDF over dependency-tree meta-structures, encouraging linguistically similar MWPs to occupy nearby regions in representation space while preserving semantics.
- We design a two-level numeracy objective to explicitly encode numerical attributes/relations and number-type cues, serving as a stabilizing regularizer that prevents representation degradation when only linguistic signals are used.
- A visual analytic scheme is proposed to elucidate how the fusion strategy enhances the mathematical representations of the encoder.

The remainder of this paper is structured as follows: Section 2 reviews related. Section 3 introduces SSN4Solver, detailing its architecture and core components. Section 4 presents the experimental results. Finally, Section 5 concludes the paper.

2. Related Work

In this section, we will introduce the main architectures of existing deep neural MWP solvers and the current problems associated with them.

2.1. Encoder–Decoder Architectures for MWPs

The development of deep neural solvers has become a main approach in MWP solving, primarily due to their capabilities in automatic feature extraction, handling complex textual information, and architectural elegance. The pioneering work [1] introduced the first deep neural solver, termed DeepNS, which employs a sequence-to-sequence (seq2seq) framework, establishing the first encoder–decoder architecture, where the encoder and the decoder are connected as a whole network. This model represents a significant departure from conventional machine learning methods. As an end-to-end solver, DeepNS utilizes Gated Recurrent Units (GRUs) for encoding and Recurrent Neural Networks (RNNs) for decoding. Following the introduction of DeepNS, the development of diverse deep neural solvers became an active research area [2,5]. Notably, Xie et al. [5] addressed limitations of traditional Seq2Seq models in MWP solving by proposing GTS, which generates solution expressions through tree structures. This advancement marked the evolution of MWP architectures toward structure-aware decoders.

The advent of BERT further revolutionized text understanding, leading to BERT-based solvers [12,13,16]. Domain-specific BERT variants, including MathBERT [12] and MWP-BERT [13], were developed to better capture mathematical relations by leveraging the semantic understanding capabilities of Pre-trained Language Models (PLMs). This progression established the foundation for PLM-based encoder–decoder architectures.

Concurrently, the relation-flow research direction has accumulated substantial experience in utilizing various representation types for MWP understanding [8,9,17]. Researchers have integrated these representations validated by relation-flow algorithms into PLM-based encoder–decoder frameworks, culminating in the use of representations to fine-tune PLM-based encoders [6,7]. Another feature of the PLM-based encoder–decoder architecture

is that its encoders are decoder-agnostic, meaning the encoder outputs can be decoded by any decoder within the scope of MWP solving.

2.2. Decoders for MWPs

Including the foundational work of DeepNS [1] discussed in our prior analysis of MWP-solving architectures, the initial encoder used a Recurrent Neural Network (RNN) framework to transform the sequential representation of math word problems (MWPs) into solution vectors. This sequence-to-sequence approach, while pioneering in its end-to-end design, exhibited inherent limitations in solution generation. Specifically, the left-to-right decoding mechanism is hard in capturing the hierarchical reasoning patterns characteristic of human problem-solving, frequently resulting in syntactically correct but semantically invalid mathematical expressions [1,2].

These limitations became particularly evident in our earlier discussion of structural decoder evolution, where the researchers noted that traditional sequence decoders struggled with the tree-like dependencies inherent in arithmetic operations. To address this challenge, subsequent research introduced tree-structured decoders such as the goal-driven tree-structured decoder (GTS) [5], which we previously identified as a key advancement in MWP architecture development. GTS innovatively maintains a goal vector throughout the decoding process, enabling it to explicitly model the hierarchical structure of mathematical expressions.

As highlighted in our comparative analysis of MWP solvers [17], GTS demonstrated superior performance in maintaining solution validity while preserving mathematical rigor. This empirical success established GTS as the frequently used decoder in recent MWP systems. Consequently, the proposed architecture in this paper adopts GTS as its decoder.

2.3. Encoders for MWPs

Encoder design presents challenges that are equally critical and often more complex than decoder development in math word problem (MWP) solvers. Early approaches involved custom deep neural network architectures [1,2,5], where researchers independently designed encoder components to process mathematical text. These initial efforts laid the groundwork for subsequent advancements in MWP representation learning.

The emergence of BERT revolutionized this landscape, prompting the development of PLM-based encoders [12,13,16]. This innovation led to domain-specific variants, including MathBERT [12] and MWP-BERT [13], which were specifically optimized to capture mathematical relations and semantic structures through enhanced contextual understanding. Subsequent research focused on enhancing PLM-based encoders through the integration of mathematically meaningful representations. Wang et al. developed MathEncoder [18], which preserves rich real-world knowledge while generating high-quality semantic representations. The model employs Template-based Contrastive Distillation Pretraining (TCDP), incorporating mathematical logic knowledge via multiview contrastive learning. This approach uses the diverse representations that are popularly used in relation-flow algorithms, which demonstrated the importance of structured mathematical representations for problem understanding [8,19].

Building on these insights, Liang et al. proposed MWP-BERT [13], a pre-trained language model that explicitly integrates numerical reasoning into pre-training. Through numeracy-augmented representation learning, MWP-BERT encodes numerical features into vector representations. Rather than treating numbers as isolated symbols, it injects numerical properties into contextualized representations, enabling the model to identify reusable numerical patterns (e.g., recognizing “10% of total” as representing proportional relations). This method addresses a key limitation of traditional encoders identified in the

researchers [6,7,20]. Lin et al. [20] developed a Relation-Enhanced Hierarchical Math Solver (RHMS) featuring a novel encoder that mimics human reading patterns through a hierarchical “word-clause-problem” structure. This approach captures inter-word dependencies for comprehensive problem understanding, utilizing graph representation learning to associate related MWPs and facilitate knowledge transfer. The work aligns with our examination of tree-structured decoders, which revealed the importance of hierarchical modeling for mathematical expressions. Tao et al. [7] introduced the Number-Centric Syntactic-Semantic Graph (NC-SSG) to address long-distance dependencies between numbers and contextual elements. The NC-SSG reconstructs dependency trees with number nodes as roots and other nodes as leaves, maintaining direct numerical-context connections while incorporating semantic number information via attention mechanisms. This methodology improves numerical context capture, reduces parsing error impacts, and enhances local structural representation, a progression we have observed in our review of encoder development.

Consequently, encoder development has evolved to focus on methods for integrating structured mathematical representations and domain-specific knowledge into large language model architectures. Inspired by the demonstrated effectiveness of mathematical structure representations in relation-flow algorithms, contemporary research emphasizes fusing syntax, semantics, and numeracy representations to build enhanced PLM-based encoders. Building upon these advances, this paper proposes SSN4Solver, which uses syntax, semantic, and numeracy representations and employs a dual learning strategy to optimize encoder training. The corporation of PLMs with contrastive learning objectives has further improved representation quality, demonstrating the synergistic potential of these complementary approaches in advancing state-of-the-art language understanding systems.

3. Building SSN4Solver by Syntax-Semantics-Numeracy Fusion

This section builds SSN4Solver by fusing syntax, semantics, and numeracy representations. We first present the overall architecture, followed by a detailed description of each component.

3.1. The Overall Architecture of SSN4Solver

This paper introduces an enhanced deep neural solver for MWP solving. The proposed model adopts an encoder–decoder architecture, utilizing a PLM as the base of the encoder and a goal-driven tree-structured (GTS) [5] model as the decoder. The research primarily focuses on improving the encoder component through a novel enhancement strategy: strengthening the encoder by fusing three distinct types of representations and using them to enhance the encoder.

In order to instill numeracy and learn general patterns, all numeric values in the problem text are replaced with a special placeholder token to create a generalized representation. This tokenization approach is essential because assigning unique tokens to each individual number would be computationally impractical and would hinder the model’s ability to capture the underlying mathematical relations between values. The primary objective of our approach is to infer these numerical relations and their numeracy directly from the contextual information provided in the problem text, rather than relying on the specific numerical values themselves.

As Figure 2 shows, SSN4Solver adopts an encoder–decoder architecture, with the decoder employing the goal-driven tree-structured decoder (GTS). The upper portion of Figure 2 outlines the component chain of the proposed solver. Following the encoder–decoder framework, the input problem P undergoes preprocessing, where numerical quantities are replaced with specialized placeholders, yielding a quantity set N_p . Subsequently, a pre-trained language model (this paper uses BERT [21] or RoBERTa [22] interchangeably)

encodes the MWP into a vector representation. The lower portion of Figure 2 depicts the structure of SSN-Encoder and its core encoder components. SSN-Encoder extracts the training loss based on three distinct feature types: (1) $\mathcal{L}_{syntax}$, representing dependency relations from syntactic parsing; (2) $\mathcal{L}_{POS}$, capturing part-of-speech information; and (3) $\mathcal{L}_{numeracy}$, encoding numerical relations. Corresponding loss functions are defined for each representation type to guide model optimization. Through continued pre-training that jointly leverages these three representation types, we obtain the fully trained SSN-Encoder.

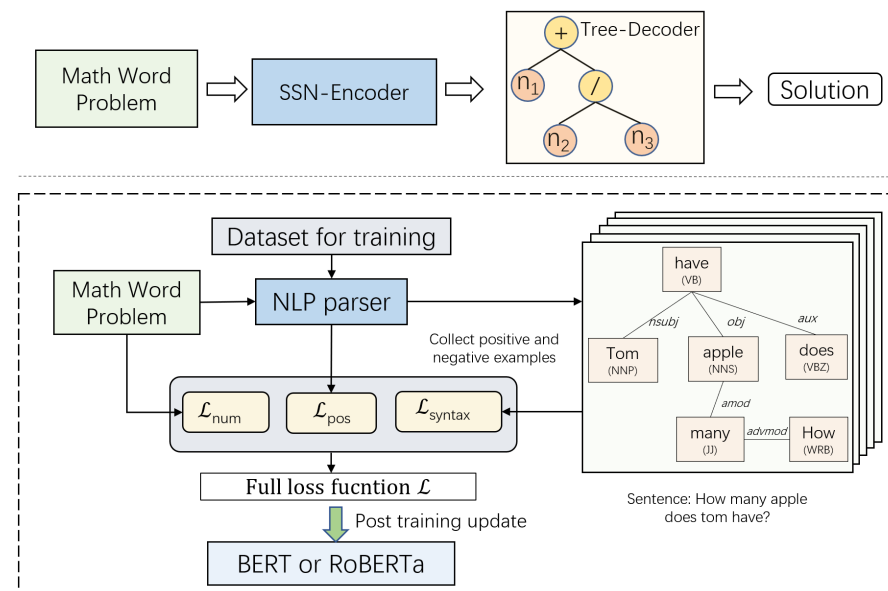


Figure 2. The component chain of the proposed deep neural solver and the component structure of its encoder. The upper portion is the proposed encoder–decoder architecture; the lower portion is the component structure of the proposed encoder.

3.2. Syntax–Semantics–Numeracy Fusion

To embed the identified three types of representation into the encoder, we first define the corresponding loss functions for them and assemble the full loss function.

3.2.1. Loss Function for Syntax

The loss function for Syntax is defined on syntax similarity measurement. We share the same view as Lin et al. that MWPs with similar syntactic structures are likely to follow analogous solution templates [20]. Consequently, the procedure for measuring the syntax similarity between two problems is outlined below, consisting of three key steps:

1. Dependency parsing: Each problem P is parsed into a dependency tree T_P that captures its syntactic structure. And many tools [23,24] are available for dependency tree construction across different languages. In the experiment, we used the Stanford CoreNLP tool [23] for parsing.
2. Meta-structure extraction: The meta-structure set M_P is extracted from each dependency tree, representing the core syntactic patterns.
3. Similarity computation: The syntax similarity of two trees are calculated using the term frequency–inverse document frequency (TF–IDF) algorithm [25] applied to the corresponding two meta-structure sets. TF–IDF is conventionally used as a token-based tool for analyzing document similarity. In our approach, we leverage the dependency trees of problem statements and extract meta-paths from these trees to serve as ‘tokens’ for TF–IDF calculation.

Figure 3 shows that every anchor problem can classify a sampled problem as a positive or negative sample. We employ the TF-IDF algorithm [25] to compute the similarity scores between different MWPs. TF-IDF encodes each document as a vector whose term weights increase with their frequency in that document but decrease with how common they are across the corpus. Using these TF-IDF vectors with cosine similarity provides a simple yet widely adopted method for computing document similarity in information retrieval and text mining.

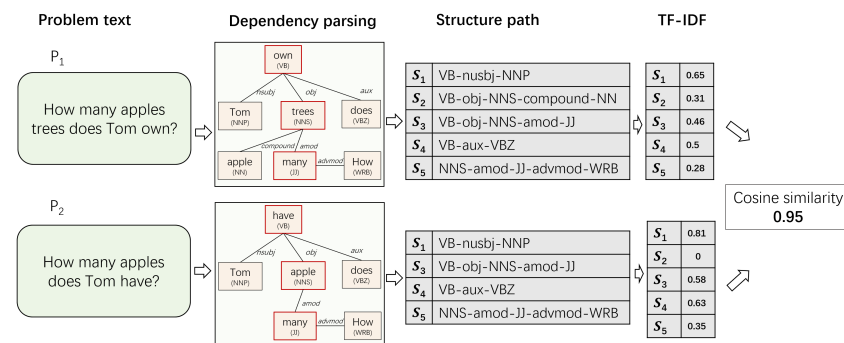


Figure 3. An example that illustrates the process of computing the structural similarity based on tf-idf with meta-structures.

Moreover, through systematic sampling, each anchor problem can explicitly identify its own set of positive and negative samples.

- Positive samples: $P_1^+, \dots, P_K^+$ —problems with similarity scores above a predefined threshold σ .
- Negative samples: $P_1^-, \dots, P_L^-$ —randomly selected problems with dissimilar dependency trees.

Based on the contrastive sampling strategy described above, the contrastive loss function can be defined. Following the approach outlined in [26], the contrastive loss for an anchor point p , denoted as

$$\mathcal{L}_{\text{InfoNCE}}^{(p)} = -\log \frac{A_+}{A_+ + B_-} \quad (1)$$

$$A_+ = \sum_{i=1}^K \exp(f(h, h_i^+)/\tau) \quad (2)$$

$$B_- = \sum_{j=1}^L \exp(f(h, h_j^-)/\tau) \quad (3)$$

where h , h_i^+ , and h_j^- denote the representation vectors of the anchor problem, positive samples, and negative samples, respectively; $f(u, v) = u^T v / (\|u\| \|v\|)$ is cosine similarity; τ is a temperature hyperparameter; and L and K are hyperparameters that are responsible for controlling the number of positive and negative samples.

For a batch of N problems, the overall syntactic-similarity regularization loss is computed as the average across batch anchors.

$$\mathcal{L}_{\text{syntax}} = \frac{1}{N} \sum_{p=1}^N \mathcal{L}_{\text{InfoNCE}}^{(p)} \quad (4)$$

3.2.2. Loss Function for Semantic POS

The loss function for Semantic POS is defined below:

$$\mathcal{L}_{pos} = \sum_{w_i \in P} CE(FFN(w_i), pos_i) \quad (5)$$

Here, w_i denotes the whole word embedding of a given mathematical word problem P after processing by a pre-trained language model encoder. pos represents part-of-speech labels, FFN is a feedforward network capturing word semantics, and CE is the cross-entropy loss.

3.2.3. Loss Function for Numeracy

To enhance the encoder's ability to capture numeracy in problem encoding, we draw inspiration from the MWP-BERT [13] framework, which introduces a series of targeted training tasks designed to improve the encoder's mathematical reasoning abilities. Concretely, we introduce two categories of optimization objectives:

- Token-level objectives: Focus on individual numerical tokens and their semantic properties.
- Sentence-level objectives: Address numerical relations across complete problem contexts.

The loss function for the token-level training objective is formally defined below:

$$\mathcal{L}_t = \sum_{w_t \in P} L(f(w_t), y_t) \quad (6)$$

Here, w_t represents the encoder output for a specific token, and f transforms the input to align with the target y in the representation space. A multilayer feedforward network is a typical example of such a transformation. The loss function L (e.g., cross-entropy loss or mean-squared error) measures the discrepancy between predictions and targets.

The token-level training objectives developed for MWPs are specifically designed to focus on numeric tokens that appear within problem descriptions. These objectives aim to establish meaningful connections between contextual number representations and their corresponding numerical properties, thereby enhancing the model's capacity to process quantitative information effectively. For example, number-type grounding is to recognize contextual cues that indicate whether a quantity represents a discrete count (such as the number of students in a classroom) or a continuous measurement (such as the distance traveled or time elapsed).

The sentence-level training objectives leverage the encoder's comprehensive understanding of a complete MWPs text description by aggregating contextual information from all constituent tokens into a unified semantic representation. The mean pooling methodology employed in sentence-level objectives serves as an effective aggregation strategy. A prominent example of sentence-level training objectives is number counting prediction, which tasks the model with accurately predicting the total quantity of numeric values present within a given MWP description. This objective serves a dual purpose: it enhances the model's attention to quantitative information while establishing connections between the frequency of numeric tokens and the complexity of variable sets required for problem resolution. MWPs frequently contain distractor information or contextual numbers that do not directly contribute to the solution process. Through systematic training on counting tasks, models develop enhanced discrimination capabilities that enable them to differentiate between relevant quantitative information and extraneous numerical details. This can help models focus on the subset of numeric values that are actually required for problem solving.

The loss function for sentence-level objectives is defined as follows:

$$\mathcal{L}_s = L(f(\bar{w}), y_s) \quad (7)$$

where L and f are the same as in (6). The comprehensive numeracy enhancement loss function is defined as the sum of two components:

$$\mathcal{L}_{\text{numeracy}} = \mathcal{L}_t + \mathcal{L}_s \quad (8)$$

3.3. Full Loss Function and Training

The full loss function is computed as the sum of three specialized components, each targeting a distinct representation of mathematical word problems:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{syntax}} + \mathcal{L}_{\text{POS}} + \mathcal{L}_{\text{numeracy}} \quad (9)$$

The training procedure is structured as follows: Stage 1: Base Model Setup Pretrained language models (PLMs) are employed as the core architecture. Stage 2: Representation Enhancement PLMs undergo explicit training to integrate three key representations of mathematical problems: syntactic structures, semantic tags, and numerical properties. In our current experimental setup, we treat the three loss components as equally important.

3.4. Remarks

Recently, a growing line of works has incorporated contrastive learning into the task of MWP solving [11,18,27]. In contrast to these approaches, our approach diverges from these existing methods in a fundamental way: whereas prior work typically constructs positive and negative sample pairs directly from the structural properties of solution expressions, we argue that the representational geometry induced by equation-level similarity is substantially more constrained than the rich semantic space underlying natural-language problem statements. Consequently, optimizing the encoder under expression-based contrastive objectives may require substantial shifts in the learned representations, potentially corrupting the semantic information that is crucial for understanding problem context and narrative structure—ultimately leading to degraded performance. Grounded on this observation, our method instead leverages syntactic structures and part-of-speech annotations to guide representation learning. Specifically, we encourage problems with linguistically similar phrasings to occupy nearby regions in the representation space, while simultaneously employing numerical features to maintain salient distinctions within semantically related problem classes. This dual-level design yields an encoder space that is more tightly coupled to both the linguistic patterns and quantitative characteristics inherent in MWP, thereby better supporting downstream solution generation.

4. Experimental Evaluation

This section evaluates the proposed SSN4Solver against 16 established baseline methods across two widely used benchmark datasets. Following conventional evaluation protocols, we first compare the overall solution accuracy of SSN4Solver with all baseline methods. Subsequently, we conduct ablation studies to quantitatively assess the contributions of different representation components within our framework. Finally, we provide visualizations to analyze how three types of mathematical representations influence the distributions of encoded vectors.

4.1. Experimental Setup

This study employs two widely used benchmark datasets for comprehensive evaluation:

- **Math23K**: A Chinese-language dataset comprising 23,164 elementary-school-level mathematical word problems, primarily focused on solving single-unknown linear equations, which was used in [1,5,11,13,20,27–31]. Each problem is annotated with its corresponding equation template, tokenized text, and numerical solution. Following the standard partitioning scheme adopted in prior work [13], the dataset is divided into 21,161 training samples, 1000 validation samples, and 998 test samples.
- **MathQA**: An English-language mathematical reasoning dataset containing 37,200 problems, which was used in [5,11,13,29–31]. Each problem is accompanied by a fully specified operational program that delineates step-by-step solution procedures, natural language rationales, and an annotated formula specifying computational steps. The dataset is split into 29,837 training samples, 4475 validation samples, and 2985 test samples.

Both datasets provide rich annotations for numerical entities, operations, and logical dependencies, making them well-suited for evaluating our syntax–semantics–numeracy fusion approach.

The proposed SSN4Solver is compared with sixteen strong baseline methods proposed in the past decade. As shown in Table 1, the selected baseline solvers are divided into three types: Seq2Seq, Seq2Tree, and Graph2Tree.

Table 1. Statistics summary for Math23K and MathQA.

Dataset	Math23K	MathQA
Num. of problem	23,162	37,259
Avg. problem length	29	37.9
Avg. number of ops	2.28	5.3
Num. of vocab	2574	6912
Num. of syntax metapath	8048	25,081
Proportion of related problems	0.24	0.86
Avg. number of relations	2.87	3.67

- **Seq2Seq** methods, such as DeepNS [1], Math-EN [28], Group-ATT [32], and T-RNN [18], use recurrent neural networks to generate mathematical expressions directly from the problem text. This approach is both intuitive and effective.
- **Seq2Tree** methods include S-Aligned [29], AST-Dec [33], HMS [34], NSSolver [35], Seq2DAG [30], and GTS [5]. These models use data structures like stacks or trees to build math expressions from the bottom up, which helps the model better understand the logical structure of the expressions. Some papers, like Graph2Tree [31] and RHMS [20], use graph-based residual connections to capture long-range relations in the problem text but still use the sequence-based encoder like GRU or LSTM.
- **PLM-based** advantages methods, such as MWP-BERT [13] and BERT-CL [11], leverage the pre-trained language models (e.g., BERT and RoBERTa), which demonstrate remarkable advantages in semantic understanding. PLMs training on vast amounts of text data through tasks like masked language modeling learn deep linguistic patterns, including syntax and context-aware meaning. The entries BERT-GTS and RoBERTa-GTS in Table 2 refer to variants where the GRU encoder is substituted with BERT to enhance semantic understanding. The PLM helps the model create more meaningful representations of the math problems.

Some operational details of the training process are described as follows: For the encoder components (BERT and RoBERTa), we adopt standard configurations with 12 trans-

former layers, 12 attention heads, and 768-dimensional hidden representations. The optimization process employs the Adam optimizer [36] with an initial learning rate of 5×10^{-5} , which undergoes scheduled reduction by half every 30 epochs. To mitigate overfitting, we apply a dropout rate of 0.5 during training, while utilizing beam search with a size of 5 for generation. For the decoder and fine-tuning procedures, we maintain consistency with the configurations reported in [13], with one notable exception: we reduce the batch size to 16 during pretraining due to the computational constraints of constructing positive examples within each batch. These positive examples are identified through similarity-based selection, where problem pairs exceeding a similarity threshold of $\alpha = 0.85$ are considered positive, while all other examples in the mini-batch serve as negative samples for contrastive learning. This approach optimizes memory usage while maintaining effective training dynamics.

Table 2. Accuracy comparison between SSN4Solver and sixteen baseline solvers on Math23K and MathQA.

	Model	Accuracy	
		Math23K	MathQA
Seq2seq	DeepNS [1]	58.1	–
	Math-EN [28]	66.7	–
	T-RNN [18]	66.9	–
	GroupAttn [32]	69.5	–
Seq2tree	S-Aligned [29]	67.1	71.3
	AST-Dec [33]	69.5	–
	NS-Solver [35]	75.6	–
	Seq2DAG [30]	72.5	75.5
	HMS [34]	76.1	–
	RHMS [20]	78.6	–
	GTS [5]	75.6	71.3
	Graph2Tree [31]	77.4	72.0
PLM	BERT-GTS	83.8	75.1
	RoBERTa-GTS	83.5	75.3
	BERT-CL [11]	83.2	76.3
	MWP-BERT [13]	84.7	76.2
	MWP-RoBERTa [13]	84.5	76.6
LLM	GPT-3.5 (Zero-shot) [37]	36.99	–
	GPT-3.5 (Zero-shot CoT) [37]	57.91	–
	GPT-4 (Zero-shot) [37]	43.05	–
	GPT-4 (Zero-shot CoT) [37]	78.16	–
Ours	SSN4Solver(BERT)	85.6	76.6
	SSN4Solver(RoBERTa)	86.0	76.8

4.2. Performance Comparison

This section first performs an accuracy evaluation comparing SSN4Solver against sixteen baseline methods in solving accuracy, followed by an ablation study assessing the individual contributions of its three core components.

4.2.1. Accuracy Comparison

In line with previous research, we employ two standard evaluation metrics: answer accuracy and expression accuracy. Answer accuracy verifies if the value derived from the generated expression tree aligns with the correct answer. Expression accuracy assesses whether the generated expression tree precisely matches the target expression tree. Notably,

if expression accuracy holds true for a math word problem, answer accuracy will also be true; however, the converse is not guaranteed. For instance, the generated expression " $n_0 + n_1 - n_2$ " differs from the target expression " $n_1 + (n_0 - n_2)$ ", yet both yield the same final result. In this scenario, answer accuracy is true, whereas expression accuracy is false. The accuracy evaluation compares the performance of SSN4Solver against sixteen baseline methods in terms of solving accuracy, with results systematically documented in Table 2.

The experimental results presented in Table 2 reveal several noteworthy patterns and trends, which we summarize through the following key observations:

- Our proposed method consistently outperforms all baseline models, demonstrating significant performance improvements. This suggests that the learned encoder effectively captures linguistic knowledge from mathematical word problems (MWP), thereby enhancing accuracy and providing superior hidden representations for the decoder. To validate this, we also evaluate the model's performance with frozen encoder parameters.
- The performance improvement is more pronounced on the Math23K dataset compared to MathQA. This discrepancy may stem from the fact that MathQA is derived from the AQuA dataset by replacing numerical or entity names. Consequently, within similar problem groups, keywords such as verbs and prepositions remain largely unchanged, limiting the model's ability to learn syntactic dependencies from diverse contexts. This lack of diversity hinders the effectiveness of our contrastive learning objective. For example, consider the logical opposition between "more than" and "less than." These phrases share similar part-of-speech tags and syntactic structures but dictate opposite mathematical operations. Our model is designed to learn these subtle distinctions by contrasting them in the embedding space. However, if a dataset is dominated by a single syntactic pattern, the model cannot effectively learn the relative logical relationship or a robust decision boundary. The experimental results indicate that the efficacy of our proposed method is significantly influenced by the linguistic diversity and semantic richness of the training data. The marginal improvement observed on the MathQA dataset reveals a potential limitation: our model relies on identifying nuanced syntactic dependencies, which are often obscured in MathQA due to its template-based nature and repetitive sentence structures.
- While we have included large language models (LLMs) in our comparative analysis, several important distinctions must be noted. First, the parameter scale of LLMs vastly exceeds that of models based on the BERT-GTS architecture. Furthermore, Math23k is not a standard benchmark typically used for evaluating LLMs, and mathematical reasoning has historically been a significant limitation, particularly for earlier iterations of these models. Specifically, the experimental results cited from [37] rely on the gpt-3.5-turbo-0301 and gpt-4-0314 versions. While the primary advantage of LLMs lies in their Artificial General Intelligence (AGI) capabilities—and we acknowledge that the latest state-of-the-art models likely demonstrate improved problem-solving performance. The scope and contribution of this study remain focused on the injection of external knowledge and constraints, as well as the investigation of internal annotations and solver performance.

4.2.2. Ablation Experiments

To evaluate the individual contributions of the three core components in SSN4Solver, namely, the multi-view contrastive learning (CL) mechanism, the semantic-aware representation optimization (SRO) module, and the math knowledge optimization (MKO) component—we conducted systematic ablation experiments. These studies were performed

on both the Math23K (Chinese) and MathQA (English) benchmark datasets to assess the cross-lingual robustness and generalization capability of each module.

The following model variants were compared to isolate the effect of each component:

- Full model: The complete framework integrating all three proposed components (CL, SRO, and MKO).
- w/o CL: A variant excluding the multi-view contrastive learning mechanism, designed to isolate the impact of the semantic optimization module.
- w/o SRO: A configuration without the semantic-aware representation optimization module, used to assess the contribution of contrastive learning.
- w/o MKO: A model where the math knowledge optimization component is removed, intended to evaluate the role of semantic enhancement.
- Baseline: A standard encoder–decoder architecture without any of the proposed enhancements, serving as the performance lower bound.

The results of our ablation study are systematically summarized in Table 3. As demonstrated, the full model consistently achieves superior performance, outperforming all ablated variants. These results strongly indicate that the three components are complementary and collectively enhance the mathematical reasoning capabilities of the model. Ablation studies across both BERT and RoBERTa reveal a consistent and noteworthy pattern: removing the numeracy term from the total loss incurs a substantial drop in task accuracy, which aligns with our hypothesis. This finding underscores that optimizing the encoder solely on syntactic and part-of-speech features takes the risk of collapsing the underlying semantic representations, thereby diminishing the model’s representational capacity for problem solving. The numeracy loss term thus serves as a critical regularizer, preventing representation degradation by ensuring that numerical properties remain sufficiently discriminative within the learned space. Interestingly, the impact of removing this term exhibits a more modest degradation on RoBERTa compared to BERT, a phenomenon we attribute to RoBERTa’s training on a substantially larger and more diverse corpus. The richer pre-trained representations in RoBERTa provide greater inherent robustness to the loss of numerical grounding, allowing the model to maintain a degree of quantitative sensitivity even when the numeracy objective is absent. This result suggests that the effectiveness of our numeracy-augmented contrastive approach is particularly pronounced in settings where the base model’s representational quality may be more limited, though it continues to provide consistent improvements across architectures.

Table 3. Ablation study on SSN4Solver.

	Math23K	MathQA
SSN4Solver (BERT)	85.5	76.6
-w/o Contrastive term	85.1	76.5
-w/o Pos term	85.1	76.5
-w/o Numeracy term	83.8	76.5
Our method-RoBERTa	86.0	76.8
-w/o Contrastive term	85.8	76.5
-w/o Pos term	85.7	76.7
-w/o Numeracy term	85.5	76.1

In particular, the performance gains on the MathQA (English) dataset are less pronounced than those on Math23K (Chinese). This discrepancy appears to be attributable to the dataset construction methodology: MathQA was automatically augmented from an existing corpus [38] primarily through the lexical substitution of names and numbers. This

process, unfortunately, is hard to introduce sufficiently diverse syntactic structures, thus limiting the effectiveness of our proposed approach in this particular context.

Despite the performance drops observed in the ablated models, both the w/o CL and w/o SRO variants still substantially outperform the Baseline model. This finding provides compelling evidence that each component independently contributes to overall enhancement. In summary, these ablation study results offer robust validation of our architectural design and confirm that both the multi-view contrastive learning and the semantic-aware representation optimization are integral to the success of our proposed approach for mathematical word problem solving.

4.3. Comparison on Embedding Effect of SSN-Encoder Against Vanilla Encoder

To elucidate how our proposed methods integrate diverse mathematical representations to enhance MWPs' encoded vectors, we visualize and compare the embedding spaces of SSN-Encoder and vanilla RoBERTa. t-SNE (t-Distributed Stochastic Neighbor Embedding), introduced by Maaten and Hinton (2008) in [39], is a nonlinear dimensionality reduction algorithm that projects high-dimensional data into lower-dimensional spaces (e.g., 2D or 3D) while preserving local structures. It achieves this by computing similarities between data points using t-distributions and minimizing the Kullback–Leibler (KL) divergence between high- and low-dimensional probability distributions. Despite t-SNE's success in domains like genomics, visually explainable methods for enhancing Mathematical Word Problem (MWP) solvers, which translate textual descriptions into mathematical equations—remain underexplored. MWP solvers often operate as “black-box” systems, obscuring their decision-making processes. Bridging this gap, novel visualization frameworks could integrate t-SNE-inspired techniques to elucidate feature embeddings or decision boundaries, thereby improving solver transparency, trustworthiness, and educational utility.

Encoder-Level Analysis and Representation Space Visualization: To examine differences at the overall encoder level, the representation spaces induced by the proposed encoder and the vanilla RoBERTa encoder are visualized using t-SNE dimensionality reduction in the Math23K dataset. As illustrated in Figure 4a,b, the embeddings produced by the proposed encoder exhibit notably greater dispersion throughout the representation space compared to the baseline BERT encoder. This increased scatter in the embedding space suggests that the proposed encoder possesses a stronger capacity to differentiate among problems of varying types and characteristics. Such enhanced discrimination capability facilitates improved overall model performance by creating more distinct and separable representations for heterogeneous input instances. To further quantify this observation, the isotropy of the representation space is measured using an isotropy score computed from the average pairwise cosine similarity. The isotropy metric is formally defined as
$$\text{Iso} = \frac{2}{N(N-1)} \sum_{i < j} \cos(\mathbf{x}_i, \mathbf{x}_j)$$
 where $\mathbf{x}_i$ represents individual embeddings and N denotes the total number of samples. The isotropy scores obtained from representations processed through the proposed encoder are substantially closer to 0 compared to those of the original baseline method (from -0.43431 to -0.19578). This proximity to zero indicates a more homogeneous and isotropic distribution of embeddings in the representation space, reflecting improved uniformity in directions and reduced redundancy in the learned representations. Such isotropy is theoretically advantageous, as it ensures that the embedding space utilizes the available dimensionality more efficiently, thereby enhancing the model's expressive capacity and generalization performance.

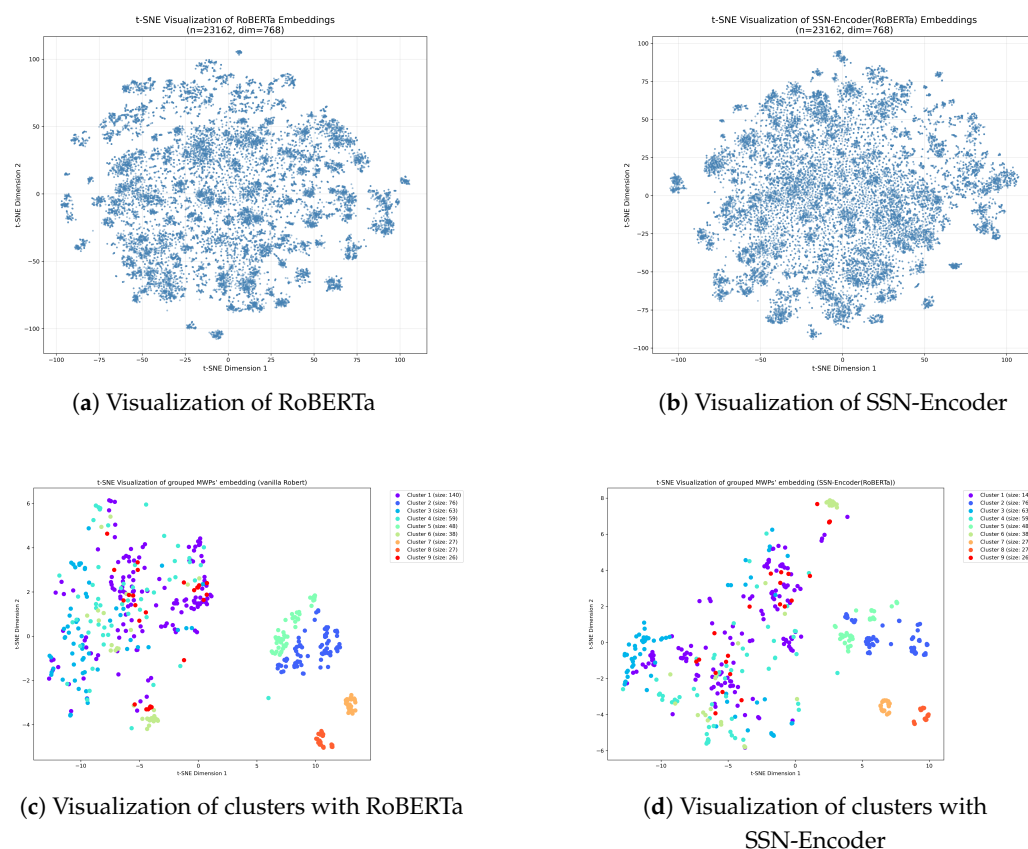


Figure 4. Visualization of the proposed SSN-Encoder versus RoBERTa. Subfigures (a,b) illustrate the embedding spaces on the Math23K dataset, while subfigures (c,d) show the evolution of clusters with similar syntactic structures.

Syntactic Structure-Based Cluster Analysis: We further investigate the evolution of instances with similar syntactic structures across the embedding space. Following the similarity algorithm described in Section 3.2, problems exhibiting distinct syntactic structures are segregated into separate clusters. As depicted in the accompanying figure, the inter-cluster distances have increased considerably, and reveal a substantial reduction in the intermingling and overlapping of clusters corresponding to different problem types. Specifically, indicating that problems with disparate syntactic structures are now mapped to more distant regions in the representation space. Our method's embeddings (Figure 4d) show more compact and cohesive clusters compared to the baseline (Figure 4c), validating our hypothesis that syntax–semantics–numeracy fusion guides semantically similar problems to proximate locations in the embedding space. Enhanced inter-class separation: For clusters 1 and 3, which overlap in the baseline, our method clearly demarcates them into distinct regions, indicating increased inter-class distance and improved discriminative bias for better problem differentiation and solver accuracy.

We further analyze the sources of the observed accuracy improvements by partitioning the test set into in-cluster and out-cluster subsets. An instance is classified as in-cluster if the training set contains problems with similar syntactic structures, whereas out-cluster instances lack such structural counterparts in the training data. This stratification allows for a granular assessment of the model's generalization capability across familiar and novel syntactic patterns.

As summarized in Table 4, the performance breakdown reveals that the accuracy gains are predominantly driven by improvements on the out-cluster instances. This enhancement suggests that the proposed method fosters a higher-quality distribution within the embed-

ding space, specifically benefiting samples that are traditionally challenging due to their distinctiveness from training examples. The results indicate that the refined representation space effectively mitigates confusion among easily conflated problem types, enabling the model to correctly answer the questions.

Table 4. Analyze the sources of the accuracy improvement.

	SSN4Solver (RoBERTa)	RoBERTa-GTS
Overall accuracy	0.85972	0.83467
In-cluster instances	220/231 (0.9524)	221/231 (0.9567)
Out-cluster instances	638/767 (0.8318)	612/767 (0.7979)

5. Conclusions and Future Work

This paper developed and validated SSN4Solver, a deep neural approach that advances MWP solving by fusing syntax, semantics, and numeracy into a unified encoder representation. In contrast to prior methods that exhibit informational asymmetry by implicitly prioritizing surface text cues, SSN4Solver explicitly treats these three sources as coequal pillars of understanding. Concretely, SSN4Solver strengthens syntactic awareness via dependency parsing, improves semantic grounding through part-of-speech supervision, and enhances numerical reasoning by modeling number-centric attributes and relational cues. Across multiple benchmark datasets, SSN4Solver consistently surpasses strong baselines, indicating that targeted representation enhancement remains a high-leverage pathway for improving neural mathematical reasoning. Beyond aggregate accuracy, our analyses provide insight into how the fused representations improve reasoning. The proposed visualization scheme shows that SSN4Solver yields more compact and cohesive representation clusters than a vanilla encoder, while simultaneously increasing separation across distinct problem types. This empirical evidence supports our core hypothesis: the enforcing and fusion procedure guides the model toward an embedding space in which structurally similar MWPs are mapped to nearby regions, facilitating generalization, and dissimilar problems remain distinguishable. Future work can extend this direction by integrating richer discourse/logic signals, improving robustness to noisy or ambiguous language, and exploring interactive or human-in-the-loop settings where transparent reasoning traces and inductive biases are essential.

Author Contributions: Conceptualization, Z.F. and X.Y.; methodology, Z.F. and H.M.; software, Z.F.; validation, H.M. and X.Y.; formal analysis, H.M.; investigation, Z.F. and H.M.; writing—original draft preparation, Z.F.; writing—review and editing, H.M. and X.Y.; supervision, X.Y.; project administration, X.Y.; funding acquisition, X.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the National Natural Science Foundation of China (62277022).

Data Availability Statement: The original data presented in the study are openly available in github at <https://github.com/SCNU203/Math23k> (accessed on 18 February 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wang, Y.; Liu, X.; Shi, S. Deep Neural Solver for Math Word Problems. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 9–11 September 2017; pp. 845–854. [CrossRef]
2. Wang, L.; Zhang, D.; Gao, L.; Song, J.; Guo, L.; Shen, H.T. MathDQN: Solving Arithmetic Word Problems via Deep Reinforcement Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.

3. Gan, W.; Yu, X.; Zhang, T.; Wang, M. Automatically Proving Plane Geometry Theorems Stated by Text and Diagram. *Int. J. Pattern Recognit. Artif. Intell.* **2019**, *33*, 1940003:1–1940003:26. [\[CrossRef\]](#)
4. Jian, P.; Sun, C.; Yu, X.; He, B.; Xia, M. An End-to-End Algorithm for Solving Circuit Problems. *Int. J. Pattern Recognit. Artif. Intell.* **2019**, *33*, 1940004:1–1940004:21. [\[CrossRef\]](#)
5. Xie, Z.; Sun, S. A Goal-Driven Tree-Structured Neural Model for Math Word Problems. In Proceedings of the International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019.
6. Zhang, Y.; Zhou, G.; Xie, Z.; Huang, J. HGEN: Learning Hierarchical Heterogeneous Graph Encoding for Math Word Problem Solving. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2022**, *30*, 816–828. [\[CrossRef\]](#)
7. Tao, X.; Zhang, Y.; Xie, Z.; Zhao, Z.; Zhou, G.; Lu, Y. Unifying the syntax and semantics for math word problem solving. *Neurocomputing* **2025**, *636*, 130042. [\[CrossRef\]](#)
8. Yu, X.; Sun, H.; Sun, C. A relation-centric algorithm for solving text-diagram function problems. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 8972–8984. [\[CrossRef\]](#)
9. Yu, X.; Cheng, W.; Yang, C.; Zhang, T. A theoretical review on solving algebra problems. *Expert Syst. Appl.* **2026**, *296*, 128789. [\[CrossRef\]](#)
10. Jian, P.; Sun, T.; Ma, B.; Xi, H.; Yang, Y. Dual Decoder Mathematical Word Problem Solving Model Based on Lie Group Intrinsic Mean Feature Matrix. *Neural Process. Lett.* **2025**, *57*, 85. [\[CrossRef\]](#)
11. Li, Z.; Zhang, W.; Yan, C.; Zhou, Q.; Li, C.; Liu, H.; Cao, Y. Seeking Patterns, Not just Memorizing Procedures: Contrastive Learning for Solving Math Word Problems. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, 22–27 May 2022.
12. Shen, J.T.; Yamashita, M.; Prihar, E.; Heffernan, N.; Wu, X.; Graff, B.; Lee, D. MathBERT: A Pre-trained Language Model for General NLP Tasks in Mathematics Education. *arXiv* **2021**, arXiv:2106.07340.
13. Liang, Z.; Zhang, J.; Wang, L.; Qin, W.; Lan, Y.; Shao, J.; Zhang, X. MWP-BERT: Numeracy-Augmented Pre-training for Math Word Problem Solving. In Proceedings of the NAACL-HLT, Seattle, WA, USA, 10–15 July 2022; pp. 997–1009.
14. Yu, X.; Lyu, X.; Peng, R.; Shen, J. Solving arithmetic word problems by synergizing syntax-semantics extractor for explicit relations and neural network miner for implicit relations. *Complex Intell. Syst.* **2023**, *9*, 697–717. [\[CrossRef\]](#)
15. Peng, R.; Yu, X.; Yang, C.; Lyu, X. A Scene-Attention Relation-Centric Algorithm for Solving Arithmetic Word Problems. *Expert Syst. Appl.* **2025**, *277*, 127197. [\[CrossRef\]](#)
16. Patel, A.; Bhattamishra, S.; Goyal, N. Are NLP Models really able to Solve Simple Math Word Problems? In Proceedings of the North American Chapter of the Association for Computational Linguistics, Online, 6–11 June 2021.
17. He, B.; Yu, X.; Huang, L.; Meng, H.; Liang, G.; Chen, S. Comparative study of typical neural solvers in solving math word problems. *Complex Intell. Syst.* **2024**, *10*, 5805–5830. [\[CrossRef\]](#)
18. Wang, L.; Zhang, D.; Zhang, J.; Xu, X.; Gao, L.; Dai, B.T.; Shen, H.T. Template-Based Math Word Problem Solvers with Recursive Neural Networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 7144–7151.
19. He, B.; Yu, X.; Jian, P.; Zhang, T. A relation based algorithm for solving direct current circuit problems. *Appl. Intell.* **2020**, *50*, 2293–2309. [\[CrossRef\]](#)
20. Lin, X.; Huang, Z.; Zhao, H.; Chen, E.; Liu, Q.; Lian, D.; Li, X.; Wang, H. Learning Relation-Enhanced Hierarchical Solver for Math Word Problems. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 13830–13844. [\[CrossRef\]](#) [\[PubMed\]](#)
21. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the North American Chapter of the Association for Computational Linguistics, Boulder, CO, USA, 1–3 June 2019; pp. 4171–4186.
22. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
23. Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; McClosky, D. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, MD, USA, 23–24 June 2014; pp. 55–60. [\[CrossRef\]](#)
24. Qi, P.; Zhang, Y.; Zhang, Y.; Bolton, J.; Manning, C.D. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Online, 5–10 July 2020; pp. 101–108.
25. Salton, G.; Yang, C.S.; Gupta, A. A Vector Space Model for Automatic Indexing. *Commun. ACM* **1975**, *18*, 613–620. [\[CrossRef\]](#)
26. van den Oord, A.; Li, Y.; Vinyals, O. Representation Learning with Contrastive Predictive Coding. *arXiv* **2018**, arXiv:1807.03748.
27. Li, Y.; Wang, L.; Kim, J.J.; Tan, C.S.; Luo, Y. On the Selection of Positive and Negative Samples for Contrastive Math Word Problem Neural Solver. In Proceedings of the 17th International Conference on Educational Data Mining, Atlanta, GA, USA, 4–17 July 2024; pp. 96–106. [\[CrossRef\]](#)

28. Wang, L.; Wang, Y.; Cai, D.; Zhang, D.; Liu, X. Translating a Math Word Problem to a Expression Tree. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 1064–1069. [[CrossRef](#)]
29. Chiang, T.R.; Chen, Y.N.V. Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems. *arXiv* **2018**, arXiv:1811.00720.
30. Cao, Y.; Hong, F.; Li, H.; Luo, P. A Bottom-Up DAG Structure Extraction Model for Math Word Problems. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021; pp. 39–46.
31. Zhang, J.; Wang, L.; Lee, R.K.-W.; Bin, Y.; Wang, Y.; Shao, J.; Lim, E.-P. Graph-to-tree learning for solving math word problems. In Proceedings of the Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 3928–3937.
32. Li, J.; Wang, L.; Zhang, J.; Wang, Y.; Dai, B.T.; Zhang, D. Modeling Intra-Relation in Math Word Problems with Different Functional Multi-Head Attentions. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 6162–6167. [[CrossRef](#)]
33. Liu, Q.; Guan, W.; Li, S.; Kawahara, D. 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), Hong Kong, China, 3–7 November 2019; pp. 2370–2379. [[CrossRef](#)]
34. Lin, X.; Huang, Z.; Zhao, H.; Chen, E.; Liu, Q.; Wang, H.; Wang, S. HMS: A hierarchical solver with dependency-enhanced understanding for math word problem. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021.
35. Qin, J.; Liang, X.; Hong, Y.; Tang, J.; Lin, L. Neural-Symbolic Solver for Math Word Problems with Auxiliary Tasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 5870–5881. [[CrossRef](#)]
36. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
37. Kim, J.; Kim, Y.; Baek, I.; Bak, J.; Lee, J. It Ain't Over: A Multi-aspect Diverse Math Word Problem Dataset. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023; pp. 14984–15011. [[CrossRef](#)]
38. Ling, W.; Yogatama, D.; Dyer, C.; Blunsom, P. Program Induction by Rationale Generation: Learning to Solve and Explain Algebraic Word Problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, BC, Canada; Computational Linguistics: Stroudsburg, PA, USA, 2017; pp. 158–167. [[CrossRef](#)]
39. Maaten, L.v.d.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.