



# City Research Online

## City St George's, University of London

**Citation:** Sundaram, S. S., Gurajada, S., Padmanabhan, D., Abraham, S. S. & Fisichella, M. (2024). Does a language model “understand” high school math? A survey of deep learning based word problem solvers. WIREs Data Mining and Knowledge Discovery, 14(4), e1534. doi: 10.1002/widm.1534

This is the published version of the paper.

This version of the publication may differ from the final published version. To cite this item please consult the publisher's version.

**Permanent repository link:** <https://openaccess.city.ac.uk/id/eprint/36024/>

**Link to published version:** <https://doi.org/10.1002/widm.1534>

**Copyright and Reuse:** Copyright and Moral Rights remain with the author(s) and/or copyright holders. Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge, unless otherwise indicated, provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way. For full details of reuse please refer to [City Research Online policy](#).

# Does a language model “understand” high school math? A survey of deep learning based word problem solvers

Sowmya S. Sundaram<sup>1</sup>  | Sairam Gurajada<sup>2</sup> | Deepak Padmanabhan<sup>3</sup> |  
Savitha Sam Abraham<sup>4</sup> | Marco Fisichella<sup>1</sup> 

<sup>1</sup>L3S Research Center, Hannover, Germany

<sup>2</sup>IBM Research, Almaden, USA

<sup>3</sup>Queen's University, Belfast, UK

<sup>4</sup>University of Örebro, Örebro, Sweden

## Correspondence

Sowmya S. Sundaram, L3S Research Center, Hannover, Germany.

Email: [sowmyassundaram@gmail.com](mailto:sowmyassundaram@gmail.com)

**Edited by:** Tianrui Li, Associate Editor and Witold Pedrycz, Editor-in-Chief

## Abstract

From the latter half of the last decade, there has been a growing interest in developing algorithms for automatically solving mathematical word problems (MWP). It is a challenging and unique task that demands blending surface level text pattern recognition with mathematical reasoning. In spite of extensive research, we still have a lot to explore for building robust representations of elementary math word problems and effective solutions for the general task. In this paper, we critically examine the various models that have been developed for solving word problems, their pros and cons and the challenges ahead. In the last 2 years, a lot of deep learning models have recorded competing results on benchmark datasets, making a critical and conceptual analysis of literature highly useful at this juncture. We take a step back and analyze why, in spite of this abundance in scholarly interest, the predominantly used experiment and dataset designs continue to be a stumbling block. From the vantage point of having analyzed the literature closely, we also endeavor to provide a road-map for future math word problem research.

This article is categorized under:

Technologies > Machine Learning

Technologies > Artificial Intelligence

Fundamental Concepts of Data and Knowledge > Knowledge Representation

## KEYWORDS

automated word problem, deep learning, natural language processing, solving

## 1 | INTRODUCTION

Natural language processing has been one of the most popular and intriguing AI-complete sub-fields of artificial intelligence. One of the earliest natural language processing systems arguably was the PhD Thesis on automatically solving math word problems (MWPs) (Bobrow, 1964). This thesis explored how to map natural language text to a set of pre-

Work done while Sowmya S. Sundaram, worked at L3S Research Center, Hannover, Germany, Sairam Gurajada worked at IBM Research, Almaden, USA and Savitha Sam Abraham worked at University of Örebro, Sweden.

This is an open access article under the terms of the [Creative Commons Attribution-NonCommercial-NoDerivs](https://creativecommons.org/licenses/by-nc-nd/4.0/) License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2024 The Authors. *WIREs Data Mining and Knowledge Discovery* published by Wiley Periodicals LLC.

defined mathematical templates and designed efficient mathematical solving methods using numerical and statistical analysis. In a similar vein, research has exploded attempting to bridge the gap of semantic parsing. In this survey paper, our topic is to review advances in the semantically involved task of solving arithmetic word problems using natural language processing methodologies, especially those in the realm of deep learning and large language models based methods.

## 1.1 | Math word problem (MWP) solving

A math word problem (MWP) is a linguistic description of a situation that calls for reasoning between mathematical entities and the application of both linguistic and mathematical knowledge to answer questions based on the situation. The complexity of math reasoning can range anywhere from elementary school level problems to complex probabilistic reasoning. The challenge lay on two fronts (a) analyzing unconstrained natural language, and (b) mapping intricate text patterns onto a small mathematical vocabulary, for usage within its reasoning framework. A typical example is given below.

|        |                                                                             |
|--------|-----------------------------------------------------------------------------|
| Input  | Kevin has 3 books. Kylie has 7 books. How many books do they have together? |
| Answer | 10                                                                          |

## 1.2 | Impact of MWP solving research

Math word problem solving is but an application of knowledge-aware semantic parsing. Semantic parsing is one of the hard challenges of Natural Language Understanding (NLU) and there exist many ways of formulating the task. Parsing semantics, or meaning, from text can involve a wide range of topics such as common sense reasoning, word sense disambiguation and so on. Playing with the complexity of this task within a contained framework such as math word problem solving is exciting for the NLP community to pursue as the methods discovered for this task can well be extrapolated to general question answering and semantic parsing frameworks. While the challenge in MWP solving is to map linguistic structures onto mathematical definitions, the general NLU task is to identify the domain relevant to the question and accordingly craft the answer within that framework. The central idea is to develop algorithms that deal with complex domains. The other application of robust and explainable MWP solving is to develop intelligent tutoring systems that can aid the learning of mathematical concepts in a minimally interventionist manner. These factors have been responsible for the explosion in interest in automated MWP solving in the last decade.

## 1.3 | Evolution of MWP solving research

Right up until 2010, there has been prolific exploration of MWP solvers, for various domains (such as algebra, percentages, ratio etc.). These solvers relied heavily on hand-crafted rules for bridging the gap between language and the corresponding mathematical notation. As can be surmised, these approaches, while being effective within their niches, did not generalize well to address the broader problem of solving MWPs. Moreover, due to the lack of well accepted datasets, it was hard to measure the relative performance across proposed systems (Mukherjee & Garain, 2008).

The pioneering work by Kushman et al. (2014) employed statistical methods to solve word problems, which set the stage for the development of automatic MWP solvers using traditional machine learning methods. The work also introduced the first dataset, popularly referred to as Alg514, that had multiple linear equations associated with a problem. The machine learning task was to map the coefficients in the equation to the numbers in the problem. The dataset comprises data units with a triplet structure: natural language question, equation set, and the final answer.

## 1.4 | Designing the survey

Mirroring recent trends in NLP, there has been an explosion of deep learning models for MWP. Some of the early ones Wang et al. (2017), Ling et al. (2017) modeled the task of converting the text to equation as a sequence-to-sequence

(seq2seq, for short) problem. In this context, increasingly complex models have been proposed to capture semantics beyond the surface text. Some have captured structural information (pertaining to input text, domain knowledge, output equation structure) in the form of graphs and used advances in graph neural networks (Li et al., 2020; Zhang, Wang, Lee, et al., 2020; etc.). Others have utilized the benefits of transformers in their modeling (Liang et al., 2021; Piękos et al., 2021; etc.). We will explore these models in detail.

Since this is a problem that has consistently attracted steady (arguably, slow and steady) attention, ostensibly right from the birth of the field of NLP, a survey of the problem solving techniques offers a good horizon for researchers. The authors collected 30+ papers on deep learning for word problem solving, published over the last 3 years across premier NLP avenues. Apart from that, the authors also analyze the state of the art Large Language Models (LLMs) applied to the word problem solving context. Each paper has its own unique intuitive basis, but most achieve comparable empirical performance. The profusion of methods has made it hard to crisply point out the state-of-the-art, even for fairly general word problem solving settings. Hence, a broad overview of the techniques employed gives a good grounding for further research. Similarly, understanding the source, settings and relevance of datasets is often important. For example, there are many datasets that are often referred to by multiple names at different points in time. Also, the finer aspects of problem scenario varies across systems (whether multiple equations can be solved, whether it is restricted to algebra or more domains etc.). In this survey, we systematically analyze the models, list the benchmark datasets and examine word problem solving literature using a critical analysis perspective.

## 1.5 | Related surveys

There are two seminal surveys that cover word problem solving research. One, Mukherjee and Garain (2008), has a detailed overview of the symbolic solvers for this problem. The second, more recent one Zhang, Wang, Zhang, et al. (2020), covers models proposed up until 2020. In the last 2 years, there has been a sharp spike in algorithms developed, that focus on various aspects of deep learning, to model this problem. Our survey is predominantly based on these deep learning models. The differentiating aspects of our survey from another related one, Faldu et al. (2021) are: the usage of a critical perspective to analyze deep learning models, which enables us to identify robustness deficiencies in the methods analytically, and also to trace them back to model design and dataset choice issues. Also, the scope of systems observed is greatly enhanced in our survey by including the latest large language model (LLM) based methods in the analysis. Our survey distinctly includes the performance values on several datasets and highlights trends that can inform insights and intuitions of the various approaches. We will also include empirical performance values of various methods on popular datasets, and deliberate on future directions.

## 2 | ORGANIZATION OF THE SURVEY

An overall bird's eye view of how the different solvers are organized is presented in Figure 1. The survey begins with the top-level categories, also encompassing a cursory description of non-neural solvers—namely symbolic and statistical solvers. After that, different types of neural solvers that employ deep learning techniques are described. This is followed by a detailed discussion on datasets used for evaluation. Following this, we analyze the performance of various systems. This is followed by a discussion on the evaluation measures. Finally, the survey ends with motivations for the future.

## 3 | MODELS

In this section, there will be a detailed description of various top-level algorithmic categories involved in developing solutions for MWPs.

### 3.1 | Non-neural solvers

We begin our discussion with solvers that do not utilize neural processing techniques. The first phase are the symbolic processors where symbols are extracted from employ a rule-based method to convert text input to a set of *symbols*.

Manipulating these symbols led to the requisite mathematical reasoning. The next set of solvers are statistical solvers that employ statistical methods to identify patterns that would, in turn, inform the mathematical reasoning.

### 3.2 | Symbolic solvers

Early solvers within this family such as STUDENT (Bobrow, 1964) and other subsequent ones (Dellarosa, 1986; Fletcher, 1985), the dominant methodology was to map natural language input to an underlying pre-defined *schema*. This calls for a mechanism to distil common expectations of language, word problems and the corresponding mathematical notation, to form bespoke rulesets that will power the conversion. This may be seen as setting up a slot-filling mechanism that map the main entities of the word problem to a slots within a set of equation templates.

An example of a schema for algebraic MWP is shown in Table 1.

The advantage is that these systems are robust in handling irrelevant information, with expert-authored rulesets enabling focus towards pertinent parts of the problem. To further enhance the practical effectiveness within applications focusing niche domains, research focused on tailoring these symbolic systems for target domains (Mukherjee & Garain, 2008).

The methods of extracting such information from the text involve simple keyword based matching, syntax parsing and could range to the complex pattern-based semantic parsing. A common trait of most systems that use this approach is to work with a specialized version of natural language, most commonly via a Controlled Natural Language. The latest in this line of work for mathematical algebraic word problems, without using empirical methods, is in Bakman (2007) which proposed a method that improved upon Dellarosa (1986). This method used “schemas” to solve addition/subtraction problems with some improvements over Fletcher (1985) for classifying entities like “dolls” as “toys.”

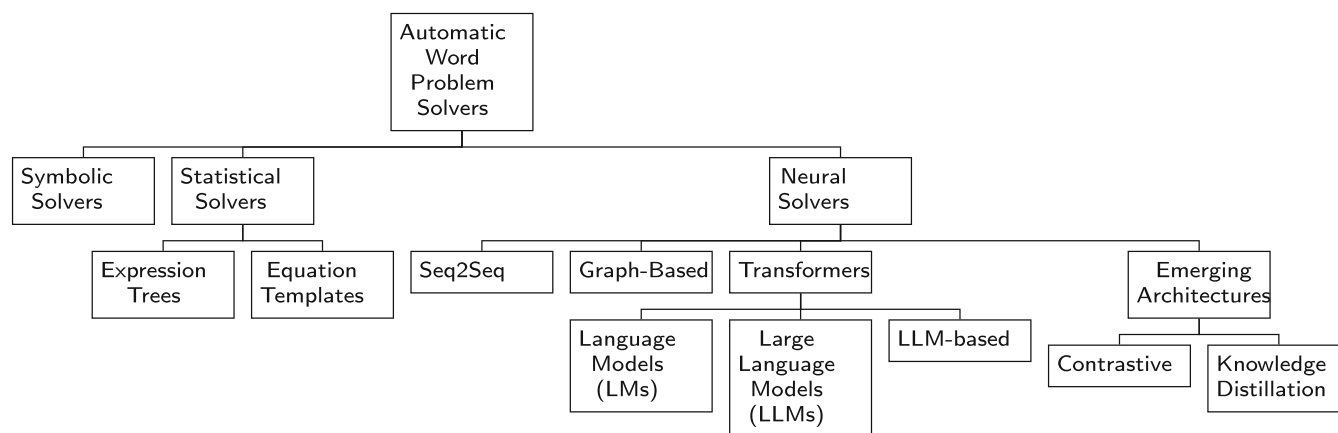


FIGURE 1 Types of word problem solvers.

TABLE 1 Workflow of symbolic solvers.

|              |                                                                                                                                                                     |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Problem      | John has 5 apples. He gave 2 to Mary. How many does he have now?                                                                                                    |
| Template     | [Owner <sub>1</sub> ] has [X] [obj].<br>[Owner <sub>1</sub> ] [transfer] [Y] [obj] to [Owner <sub>2</sub> ].<br>[Owner <sub>1</sub> ] has [Z] [obj].<br>$Z = X - Y$ |
| Slot-filling | [John] has [5] [apple].<br>[John] [give] [2] [apple] to [Mary].<br>[Mary] has [Z] [apple].<br>$Z = 5 - 2$                                                           |
| Answer       | $Z = 3$                                                                                                                                                             |

Schemas may be regarded as pre-specified templates for problem solving. Bakman (2007) extends schemas to handle extraneous information and can solve multi-step problems unlike its predecessors. Better semantic parsing was accomplished through statistical methods. The interested reader can look up the detailed review presented by Mukherjee and Garain (2008) for the detailed comparison of such systems. As one can observe, the rules would need to be exhaustive to capture the myriad nuances of language. Thus, they did not generalize well across varying language styles. Since each system was designed for a particular domain, comparative performance evaluation was hindered by the unavailability of cross-domain datasets.

### 3.3 | Statistical solvers

As with many tasks in natural language processing, statistical machine learning techniques to solve word problems started dominating the field from 2014. The central theme of these techniques has been to score a number of potential solutions (may be equations or expression trees as we will see shortly) within an optimization based scoring framework, and subsequently arrive at the correct mathematical model for the given text. This may be thought of as viewing the task as a *structure prediction challenge* (Zhang, Wang, Zhang, et al., 2020).

$$P(y|x; \theta) = \frac{e^{\theta \cdot \phi(x,y)}}{\sum_{y' \in Y} e^{\theta \cdot \phi(x,y')}} \quad (1)$$

As with optimization problems, Equation 1 refers to the problem of learning parameters  $\theta$ , which relate to the feature function  $\phi$ . Consider labeled dataset  $D$  consisting of  $n$  pairs  $(x,y,a)$  where  $x$  is the natural language question,  $y$  is the mathematical expression and  $a$  is the numerical answer. The task is to score all possible expressions  $Y$ , and maximize the choice of the labeled  $y$  through an optimization setting. This is done by modifying the parameters  $\theta$  of the feature function  $\phi(x,y)$ . Different models propose different formulations of  $\phi$ . In practice, beam search is used as a control mechanism. We grouped the prolific algorithms that were developed, based on the type of mathematical structure  $y$ —either as *equation templates* or *expression trees*. Equation templates were mined from training data, much like the slot filling idea of symbolic systems. However, they became a bottleneck to generalizability, if the word problem at inference time, was from an unseen equation template. To address this issue, expression trees, with unambiguous post-fix traversals, were used to model equations. Though they restricted the complexity of the systems to single equation models, they offered wider scope for generalizability.

#### 3.3.1 | Equation templates

Equation templates extract out the numeric coefficients and maintain the variable and operator structure. This was used as a popular representation of mathematical modeling. To begin with, Kushman et al. (2014), used structure prediction to score both equation templates and alignment of the numerals in the input text to coefficients in the template. Using a state based representation, Hosseini et al. (2014) modeled simple elementary level word problems with emphasis on verb categorization. Zhou et al. (2015) enhanced the work done by Kushman et al. (2014) by using quadratic programming to increase efficiency. Upadhyay and Chang (2017) introduced a sophisticated method of representing derivations in this space.

#### 3.3.2 | Expression trees

Expression trees are applicable only to single equation systems. The single equation is represented as a tree, with leaves of the tree being numbers and the internal nodes being operators as illustrated in Koncel-Kedziorski et al. (2015).

Expression tree based methods converge faster, understandably due to the diminished complexity of the model. Some solvers (such as Roy & Roth, 2015) had a joint optimization objective to identify relevant numbers and populating the expression tree. On the other hand, Koncel-Kedziorski et al. (2015) and Mitra and Baral (2016) used domain knowledge to constrain the search space.

## 4 | NEURAL SOLVERS

Among the major challenges for the solvers we have seen so far was that of converting the input text into a meaningful feature space to enable downstream solving; the main divergences across papers seen across the previous sections has been based on the technological flavor and methodology employed for such text-to-representation conversion.

The advent of distributed representations for text (Le & Mikolov, 2014); Peters et al., 2018; Pennington et al., 2014; Devlin et al., 2018), marked a sharp departure in the line of inquiry towards solving math word problems, focusing on the details of the learning architecture rather than feature-space modeling. There have even been domain specific distributed representation learners for word problems (Sundaram et al., 2020). As an example of solvers, Ling et al. (2017) designed a seq2seq model that incorporated learning a program as an intermediate step. This and other early works made it fashionable to treat the word problem solving task as a language translation task, i.e., translating from the input natural language text to a sequence of characters representing either the equation or a sequence of predicates. This design choice, however, has its limitations, which are sometimes severe in terms of the restrictions they place on math problems that can be admitted within such architectures (Patel et al., 2021.) A few of these linguistic vs. math structure understanding challenges, especially for neural solvers, are illustrated in Figure 2. As an important example, equation systems that involve solving multiple equations are not straightforward to address within such a framework. A notable exception to this is the popular baseline MathDQN Wang et al. (2018), which employs deep reinforcement learning. We consider different families of deep learning solvers within separate sub-sections herein.

### 4.1 | Seq2Seq solvers

The popular Seq2Seq Sutskever et al. (2014) architecture is widely popular for automatic word problem solving. The central theme here is to facilitate conversion from one sequence to another. As language is a sequence of text and mathematical reasoning can be naturally fashioned into a sequence of reasoning steps, this is a popular paradigm. The common formulations are described in Figure 3. From early direct use of LSTMs (Hochreiter & Schmidhuber, 1997)/GRUs (Cho et al., 2014) in Seq2Seq models (Huang et al., 2017; Wang et al., 2017) to complex models that include domain knowledge (Chiang & Chen, 2019; Ling et al., 2017; Qin et al., 2020; Qin et al., 2021), diverse formulations of this basic architecture have been employed. The details of these systems are presented in Table 2.

The initial set of models used Seq2Seq as is, with small variations in the usage of LSTM or GRUs or with simple heuristics (for example, Huang et al., 2016 used retrieval to enhance the results). Significant improvements were made by including some mathematical aspects. This, once again, demonstrates that the task is not merely that of language translation. Ling et al. (2017) converted the word problem to a text containing the explanation or rationale. This was done through an intermediate step of generating a step-by-step program on a large dataset. Though the accuracy values reported were low, the domains spanned anywhere between probability to relative velocity, and the unified framework demonstrated performing meaningful analysis through qualitative illustrations. This was improved upon by Amini et al. (2019), which enhanced the dataset and added domain information through a label on the category. The SAU-Solver Qin et al. (2020) introduced a tree like representation with semantic elements that align to the word problem. As seen in Table 6, this is a formidable contender. In Chiang and Chen (2019), a novel way of decomposing the equation construction into a set of stack operations—such that more nuanced mapping between language and operators can be learned—was designed.

*Interpretation* Seq2Seq solvers demonstrate, with enough data, great strides in linguistic understanding. Unlike non-deep systems, which modeled language in a brittle fashion (by counting entities and so on), the models can capture complex relationships. As the embeddings improve, the systems generally tend to show better contextual understanding (for example, identifying which quantities are relevant and which numbers need to be ignored). However, these systems have no sense of mathematical accuracy when it comes to generating numbers and equations. Often numbers that do not occur in the question appear, or generate mathematically improbable statements like (3–7 for a bunch of apples which cannot be negative). The two program based approaches Amini et al. (2019) and Ling et al. (2017) fare better in mathematical accuracy but since they model a wide variety of domains, conceptual understanding is lacking on expert examination. These qualitative analyses by the authors motivate the need for better metrics that is outlined later in this paper. In general, adding some form of domain knowledge is seen to enhance performance.

**Training Input****Sample 1**

Question Kevin has 3 books. Kylie has 7 books and 3 pencils. How many books do they have together?)

Equation1  $3 + 7$

**Sample 2**

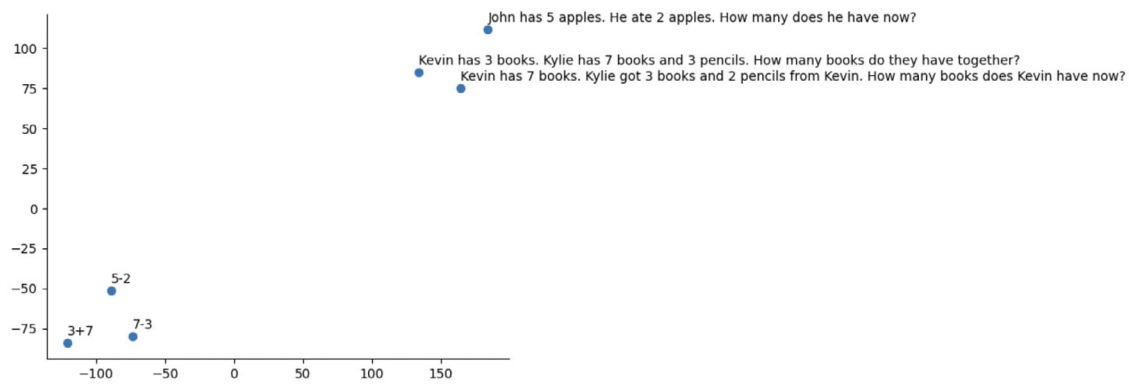
Question Kevin has 7 books. Kylie got 3 books and 2 pencils from Kevin. How many books does Kevin have now?)

Equation1  $7 - 3$

**Sample 3**

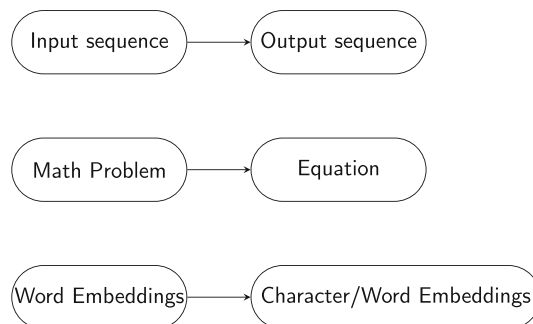
Question John has 5 apples. He ate 2 apples. How many does he have now??)

Equation1  $5 - 2$

**Vectorial Representation - GPT-2 embeddings projected onto two dimensions using t-SNE**

The vectorial representations of MWPs lie quite close to each other even though mathematical semantics may differ. Hence, multi level semantics may not be captured due to limitations in dataset design or size

**FIGURE 2** Schematic of a neural solver.



**FIGURE 3** General Seq2Seq formulations.

## 4.2 | Graph-based solvers

With the advent of graph modeling (Xia et al., 2019) and enhanced interest in multi-modal processing, the graph data structure became a vehicle for adding knowledge to solvers. One way of enabling this has been to simply model the input problem as a graph (Feng et al., 2021; Hong et al., 2021; Li et al., 2020; Yu et al., 2021). This incorporates domain knowledge of (a) language interactions pertinent to mathematical reasoning, or (b) quantity graphs stating how various numerals in the text are connected. Another way is to model the decoder side to accept graphical input of equations (Cao et al., 2021; Lin et al., 2021; Liu, Guan, et al., 2019; Wu, Zhang, Wei, & Huang, 2021; Xie & Sun, 2019; Zaporozjets

**TABLE 2** Description of Seq2Seq Solvers describing the structure of the proposed method by elaborating on the input sequence, output sequence and the inclusion of domain knowledge.

| System                 | Input sequence                  | Output sequence                         | Domain knowledge included? |
|------------------------|---------------------------------|-----------------------------------------|----------------------------|
| Huang et al. (2016)    | Word embeddings (word problem)  | Equation                                | No                         |
| Ling et al. (2017)     | Word embeddings (word problem)  | Word embeddings (rationale) and program | Yes                        |
| Amini et al. (2019)    | Word embeddings (word problem)  | Word embeddings (program)               | Yes                        |
| Chiang and Chen (2019) | Word embeddings (word problem)  | Stack operations for equation           | Yes                        |
| Qin et al. (2020)      | Tree embeddings of word problem | Word embeddings (equation)              | Yes                        |

et al., 2021). Another natural pathway that has been employed towards leveraging graphs is to use graph neural networks for both encoder and decoder (Shen & Jin, 2020; Wu et al., 2020; Wu, Zhang, & Wei, 2021; Zhang, Wang, Lee, et al., 2020). These methods of using graphs effectively are schematically represented in Figure 4.

**Interpretation** Graphs are capable of representing complex relationships. The detailed description of how exactly graphs are utilized in various systems is described in Table 3. With the time-tested success of graph neural networks (GNNs) (Wu, Pan, Chen, Long, et al., 2021), they fit easily into the encoder-decoder architecture. Intuitively, when graphs are used on the input side, we can model complex semantic relationships in the linguistic side of the task. When graphs are used on the decoder side, relationships between the numerical entities or an intermediate representation of the problem can be captured. Analogously, graph-to-graph modeling enables matching the semantics of both language and math. This does not necessarily imply graph-to-graph outperforms all the other formulations. There are unique pros and cons of each of the graph-based papers, as both language and mathematical models are hard to (a) model separately and (b) model the interactions. The interesting observation as seen in Table 6, graph based models are both popular and powerful. Unlike sequences, when the input text is represented as a graph, the focus is more on relevant entities rather than a stream of text. Similarly, quantity graphs or semantics informed graphs, eliminate ordering ambiguities in equations. This formulation, however, still does not address the multiple equation problem. On average, we expect the graph-based systems to do better than the Seq2Seq solvers, as validated in Table 6.

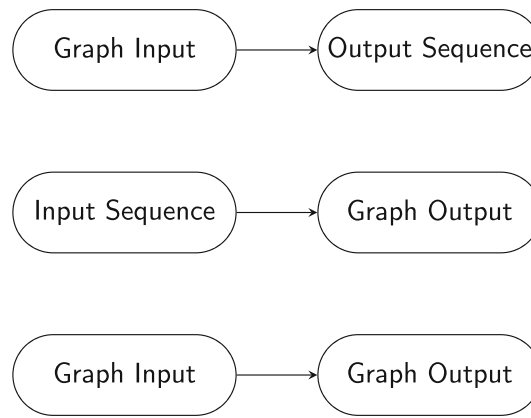
### 4.3 | Emerging architectures

Apart from these major architectures, there are upcoming different designs of neural networks that formulate the problem statement. These are grouped under emerging architectures and described below.

#### 4.3.1 | Contrastive solvers

With the widespread usage of Siamese networks (Koch et al., 2015), the idea of building representations that *contrast* between vectorial representations across classes in data has seen some interest. In the context of word problem solving, a few bespoke transformer based encoder-decoder models (Hong et al., 2021; Li, Zhang, et al., 2021) have been proposed; these seek to effectively leverage contrastive learning (Le-Khac et al., 2020).

This is a relatively new paradigm and more research needs to emerge to ascertain definite trends. One of the main stumbling blocks of word problem solving is that two highly *linguistically* similar looking word problems may have entirely different *mathematical* structure. Since contrastive learning is built on the principle that similar input examples lead to closer representations, it allows one to use the notion of similarity and dissimilarity to overcome this bottleneck and consciously design semantically informed intermediate representations, such that the similarity is built not only from the language vocabulary, but also from the mathematical concepts.



**FIGURE 4** General graph based formulations.

**TABLE 3** Description of graph-based solvers by elaborating on the input graph, output graph and the inclusion of domain knowledge.

| System                           | Input sequence                   | Output sequence                | Domain knowledge included?                       |
|----------------------------------|----------------------------------|--------------------------------|--------------------------------------------------|
| Feng et al. (2021)               | Word embeddings                  | Graph of quantities and values | Yes, on numbers and equations                    |
| Li et al. (2020)                 | Graph of quantities              | Expression tree                | Yes, quantities and numbers                      |
| Yu et al. (2021)                 | Graph of linguistic properties   | Expression tree                | Yes, language structure and numbers              |
| Hong et al. (2021)               | Word embeddings                  | Expression tree                | No                                               |
| Xie and Sun (2019)               | Word embeddings                  | Graph of quantities            | Yes, relationship between numbers and quantities |
| Lin et al. (2021)                | Word embeddings                  | Expression tree of quantities  | No                                               |
| Cao et al. (2021)                | Word embeddings                  | Dag of quantities and numbers  | Yes, between text and numbers                    |
| Zaporjets et al. (2021)          | Word embeddings                  | Expression tree                | No                                               |
| Liu, Guan, et al. (2019)         | Word embeddings                  | Expression tree                | No                                               |
| Wu, Zhang, Wei, and Huang (2021) | Word embeddings                  | Complex expression tree        | Yes, quantities and numbers                      |
| Zhang, Wang, Lee, et al. (2020)  | Quantity and text matching graph | Expression tree                | Yes, both text and numbers                       |
| Wu et al. (2020)                 | Knowledge graph of terms         | Expression tree                | Yes, both text and numbers                       |
| Wu, Zhang, and Wei (2021)        | Enhanced graph using parsers     | Expression tree                | Yes, both text and numbers                       |
| Shen and Jin (2020)              | Enhanced graph using parsers     | Complex expression tree        | Yes, both text and numbers                       |

#### 4.3.2 | Teacher-student solvers

The paradigm of knowledge distillation, in the wake of large, generic end-to-end models, has become popular in NLP (Li, Lin, et al., 2021). The underlying idea behind this is to *distill* smaller task-specific models from a generic large pre-trained or generic model. Since word problem datasets are of comparatively smaller size, it is but logical that large generic networks can be fine-tuned for downstream processing of word problem solving, as favorably demonstrated by Zhang, Lee, Lim, et al. (2020) and Hong et al. (2021).

Once again, this is an emerging paradigm. Similar to the discussion we presented with transformer based models, the fact that the presence of pre-trained language models alone is not sufficient for this task has bolstered initial efforts in this direction. Knowledge distillation enables a model to focus the learnings of one generic model on to a smaller, more focused one, especially with less datapoints. Hence, the method of adding semantic information through the usage of knowledge distillation algorithms is promising and one to look out for.

**TABLE 4** Description of transformer-based solvers by elaborating on the input graph, output graph and the inclusion of domain knowledge.

| System                     | Input sequence                 | Output sequence                                    | Domain knowledge included? |
|----------------------------|--------------------------------|----------------------------------------------------|----------------------------|
| Piękos et al. (2021)       | Word embeddings (word problem) | Word embeddings (explanation)                      | Yes                        |
| Griffith and Kalita (2020) | Word embeddings (word problem) | Word embeddings (explanation)                      | Yes                        |
| Shen et al. (2021)         | Word embeddings (word problem) | Word embeddings (equation)                         | Not explicitly             |
| Liang et al. (2021)        | Word embeddings (word problem) | Stack operations for equation                      | Yes                        |
| Jie et al. (2022)          | Word embeddings (word problem) | Complex sequence of relations                      | Yes                        |
| Zhang and Moshfeghi (2022) | Word embeddings (word problem) | Complex sequence of operator, memory and reasoning | Yes                        |

### 4.3.3 | Domain-niche solvers

Some research, encompassing families of statistical solvers and deep models, focus on the pertinent characteristics of a particular domain in mathematics, such as probability word problems (Dries et al., 2017; Suster et al., 2021; Tsai et al., 2021), number theory word problems (Shi et al., 2015), geometry word problems (Chen et al., 2021; Seo et al., 2015) and age word problems (Sundaram & Abraham, 2019).

### 4.3.4 | Neuro-symbolic solvers

There is a burgeoning section of the literature that is invested in using neuro-symbolic reasoning to bridge this gap between perception level tasks (language understanding) and cognitive level tasks (mathematical reasoning). In an end-to-end fashion, most of these systems have a combination of neural processing for language processing and symbolic solvers for the math part. An example of this is Qin et al. (2021).

### 4.3.5 | Interpretation

Emerging architectures require more research to ascertain their defining features. The preliminary research effort suggests that, for a particular subset/domain, these methods fair well. Some of them provide interesting generalizability abilities as well as seen in Table 7. With this discussion, it is clear that adding some form of domain knowledge benefits an automatic solver. These works provide some directions that may be applied to the popular transformer based paradigms as described below.

## 4.4 | Transformers

Transformers (Vaswani et al., 2017) have lately revolutionized the field of NLP. Word problem solving has been no exception. Through the use of BERT (Devlin et al., 2018) embeddings or through transformer based encoder-decoder models, some recent research has leveraged concepts from transformer models (Kim et al., 2020; Liu, Guan, et al., 2019).

### 4.4.1 | Language models

Language Models (LM) such as BERT (Devlin et al., 2018), RoBERTa (Liu, Ott, et al., 2019), T5 (Raffel et al., 2020), BART (Lewis et al., 2019), BigBird (Zaheer et al., 2020), ALBERT (Lan et al., 2019) use the transformer as a core component but vary widely in size, execution and design. The embeddings generated from these models encompassed context and were pre-trained on relatively larger pieces of text, with millions of parameters. These language models exploited

**TABLE 5** Description of datasets including their commonly used names, their descriptions and their sizes.

| Dataset                      | Type            | Domain       | Size  | Source                          |
|------------------------------|-----------------|--------------|-------|---------------------------------|
| <i>Small datasets</i>        |                 |              |       |                                 |
| Alg514 (SimulEq-S)           | Multi-equation  | (+, -, *, /) | 514   | Kushman et al. (2014)           |
| AddSub (AI2)                 | Single-equation | (+, -)       | 340   | Hosseini et al. (2014)          |
| SingleOp (Illinois, IL)      | Single-equation | (+, -, *, /) | 562   | Roy et al. (2015)               |
| SingleEq                     | Single-equation | (+, -, *, /) | 508   | Koncel-Kedziorski et al. (2015) |
| MAWPS                        | Multi-equation  | (+, -, *, /) | 3320  | Koncel-Kedziorski et al. (2016) |
| MultiArith (Common Core, CC) | Single-equation | (+, -, *, /) | 600   | Roy and Roth (2015)             |
| AllArith                     | Single-equation | (+, -, *, /) | 831   | Roy and Roth (2017)             |
| Perturb                      | Single-equation | (+, -, *, /) | 661   | Roy and Roth (2017)             |
| Aggregate                    | Single-equation | (+, -, *, /) | 1492  | Roy and Roth (2017)             |
| DRAW-1k                      | Multi-equation  | (+, -, *, /) | 1k    | Upadhyay and Chang (2017)       |
| AsDIV-A                      | Single-equation | (+, -, *, /) | 2373  | Miao et al. (2020)              |
| SVAMP                        | Single-equation | (+, -, *, /) | 1000  | Patel et al. (2021)             |
| <i>Large datasets</i>        |                 |              |       |                                 |
| Dolphin18k                   | Multi-equation  | (+, -, *, /) | 18k   | Huang et al. (2016)             |
| AQuA-RAT                     | Multiple-choice | -            | 100k  | Ling et al. (2017)              |
| Math23k*                     | Single-equation | (+, -, *, /) | 23k   | Huang et al. (2017)             |
| MathQA                       | Single-equation | (+, -, *, /) | 35k   | Amini et al. (2019)             |
| HMWP*                        | Multi-equation  | (+, -, *, /) | 5 k   | Qin et al. (2020)               |
| Ape210k*                     | Single-equation | (+, -, *, /) | 210k  | Liang et al. (2021)             |
| GSM8k                        | Single-equation | (+, -, *, /) | 8.5k  | Cobbe et al. (2021)             |
| CM17k*                       | Multi-equation  | (+, -, *, /) | 17k   | Qin et al. (2021)               |
| MATH                         | Multi-equation  | (+, -, *, /) | 12.5k | Hendrycks et al. (2021)         |

Note: Starred datasets are Chinese datasets.

the concept of *masking*, or allowing the model to *guess* a part of the sentence, and not just a single token. They also incorporated sub-word tokenizations that handled out-of-vocabulary issues of previous models. The authors could not find authentic sources of evaluation on these basic language models. With cursory exploration, they concluded that the performance was poor and brittle. However, models have been adapted, fine-tuned or utilized in neural architectures to obtain much better results, which will be described below.

#### 4.4.2 | LM based models

The translation using language models has been modeled variously in these systems, such as from text to explanation (Griffith & Kalita, 2020; Pięko et al., 2021), or from text to equation (Liang et al., 2021; Shen et al., 2021). Their performance has been listed in Table 6. Table 4 describes the method of representations.

*Interpretation* When moving from Word2Vec (Mikolov et al., 2013) vectors to BERT embeddings (Devlin et al., 2018), massive gains were expected due to (a) greater incorporation of context level information and (b) automatic capturing of relevant information as BERT is essentially a Masked Language Model. Interestingly, the gains do not have as large a margin as seen in other language tasks such as question answering or machine translation (Devlin et al., 2018). BERT is a large model that needs to be fine tuned with domain specific information. The small gains point towards low quality of word problem datasets, which is in line with the fact that the datasets are either quite small by deep learning standards or that they have high lexical overlap, effectively suggesting that the set of characteristic word problems are small.

TABLE 6 Answer accuracy of neural models evaluated on three popular datasets—Math23k, MAWPS and GSM8k.

| Model name                    | Math23k     | MAWPS       | GSM-8K      | Source                         |
|-------------------------------|-------------|-------------|-------------|--------------------------------|
| <i>Graph-based</i>            |             |             |             |                                |
| GTSs                          | 74.3        | -           | -           | Xie and Sun (2019)             |
| SAU-SOLVER                    | 74.8        | -           | -           | Chiang and Chen (2019)         |
| Graph2Tree                    | 77.4        | -           | -           | Li et al. (2020)               |
| KA-S2T                        | 76.3        | -           | -           | Wu et al. (2020)               |
| Graph-To-Tree                 | 78.8        | -           | -           | Li et al. (2020)               |
| NumS2T                        | 78.1        | -           | -           | Wu et al. (2020)               |
| Multi-E/D                     | 78.4        | -           | -           | Shen and Jin (2020)            |
| Seq2DAG                       | 77.1        | -           | -           | Cao et al. (2021)              |
| EEH-D2T                       | 78.5        | 84.8        | -           | Wu, Zhang, and Wei (2021)      |
| Generate and rank             | <b>85.4</b> | 84.0        | -           | Shen et al. (2021)             |
| HMS                           | 76.1        | 80.3        | -           | Lin et al. (2021)              |
| RPKHS                         | 83.9        | 89.8        | -           | Yu et al. (2021)               |
| <i>Emerging architectures</i> |             |             |             |                                |
| Graph-Teacher                 | 79.1        | 84.2        | -           | Liang and Zhang (2021)         |
| CL                            | 83.2        | -           | -           | Li, Zhang, et al. (2021)       |
| NS-Solver                     | 75.7        | -           | -           | Qin et al. (2020)              |
| <i>Transformer</i>            |             |             |             |                                |
| TSN-MD                        | 77.4        | 84.4        | -           | Zhang, Lee, Lim, et al. (2020) |
| EPT                           | -           | 84.5        | -           | Kim et al. (2020)              |
| Group-att                     | 69.5        | 76.1        | -           | Li et al. (2019)               |
| <i>LLMs</i>                   |             |             |             |                                |
| GPT-3                         | -           | 19.8        | 57.1        | Schick et al. (2023)           |
| LLaMa 2                       | -           | 82.4        | -           | Touvron et al. (2023)          |
| GPT-4                         | -           | -           | 92.0        | OpenAI (2023)                  |
| PaLM-2                        | -           | -           | 91.0        | Anil et al. (2023)             |
| Minerva                       | -           | -           | 58.8        | Lewkowycz et al. (2022)        |
| <i>LLM-based</i>              |             |             |             |                                |
| PaL                           | -           | -           | 92.9        | Gao et al. (2023)              |
| MsAT-DeductReasoner           | -           | <b>94.3</b> | -           | Wang and Lu (2023)             |
| EPT                           | -           | 88.7        | -           | Kim et al. (2022)              |
| MS                            | -           | -           | <b>96.8</b> | Zhao et al. (2023)             |
| PHP                           | -           | -           | 96.5        | Zheng et al. (2023)            |

Note: The best values are bolded.

## 4.5 | Large language models

Large Language Models (LLMs) have demonstrated unprecedented levels of natural language comprehension by virtue of their sheer size (often in the billions), the mechanisms of human feedback and components of both language understanding and generation. Some notable examples are PaLM (Chowdhery et al., 2022), BLOOM (Scao et al., 2022), LLaMa (Touvron et al., 2023) and so on. The most popular set of LLMs are the Generative Pre-trained Transformers or GPTs. Using a combination of mammoth size, autoregressive model and human feedback, GPTs have consistently shown massive improvements. GPT-2 (Radford et al., 2019), GPT-3 (Brown et al., 2020; OpenAI, 2023) performance of these LLMs are depicted in Table 7. Unlike other models, the authors have placed source as where this particular model was evaluated, rather than the development of the model. A mathematical LLM has been developed (Lewkowycz et al., 2022) using mathematical data. There has been word problem evaluation using these methods as well.

**TABLE 7** Analysis of large multi-domain datasets AQuA-RAT and MathQA on the performance of available models.

| Model         | Type        | AQuA-RAT    | MathQA      | Source                     |
|---------------|-------------|-------------|-------------|----------------------------|
| AQuA          | Seq2Seq     | 36.4        | -           | Ling et al. (2017)         |
| Seq2Prog      | Seq2Seq     | <b>37.9</b> | 57.2        | Amini et al. (2019)        |
| BERT-NPROP    | Transformer | 37.0        | -           | Piękos et al. (2021)       |
| Graph-To-Tree | Graph-based | -           | 69.7        | Li et al. (2020)           |
| RelExt        | Transformer | -           | 78.6        | Jie et al. (2022)          |
| ELASTIC       | Transformer | -           | <b>83.0</b> | Zhang and Moshfeghi (2022) |

Note: The best values are bolded.

## 4.6 | LLMs based

Some models have been developed that use LLMs (Gao et al., 2023; Kim et al., 2022; Schick et al., 2023; Wang & Lu, 2023; Xie et al., 2023; Zhao et al., 2023; Zheng et al., 2023) as their core components. Some facets of these systems are described below.

### 4.6.1 | Prompt engineering

Prompt engineering is the task of use zero shot or few shot learning using cleverly chosen prompts that coaxes the implicit knowledge present in the large pre-trained models.

### 4.6.2 | Chain-of-thought (CoT) prompting

Converting a query into a set of intermediate queries in a sequential manner to elicit better reasoning from LLMs.

### 4.6.3 | Program-aided language modeling

Program Aided Language Modeling (Gao et al., 2023) involves capitalizing on the code completion aspect of language models to enhance reasoning capacities.

### 4.6.4 | Adapter tuning

Adapter Tuning (He et al., 2021) is a light-weight response to fine-tuning where the entire model is re-trained. In this method, the last few layers alone are available for weight updating.

*Interpretation* LLMs and LLM-based models offer the best performance as evidenced by Table 6 and Table 8. Their size of training data and parameter size, intelligent design and human feedback incorporate common sense knowledge and mathematical domain knowledge that help push these models to a much better performance.

## 5 | DATASETS

Datasets used for math word problem solving are listed in Table 5 with their characteristics. The top section of the table describes datasets with relatively fewer data objects ( $\leq 1k$ , to be specific). The bottom half consists of more recent datasets that are larger and more popularly used within deep learning methodologies.

**TABLE 8** Maximum performance of different types of neural models on three popular datasets—Math23k, MAWPS and GSM8k are explored.

| Type                   | Max. Accuracy on Math23k | Max. Accuracy on MAWPS | Max. Accuracy on GSM8k |
|------------------------|--------------------------|------------------------|------------------------|
| Graph-based            | 85.4                     | 89.80                  |                        |
| Seq2Seq                | 75.67                    | -                      |                        |
| Transformer based      | 69.50                    | 84.5                   |                        |
| Emerging Architectures | 83.2                     | 84.4                   |                        |
| LLMs                   | -                        | <b>94.3</b>            | <b>96.8</b>            |

Note: The best values are bolded.

## 5.1 | Small datasets

The pioneering work in solving word problems (Kushman et al., 2014), introduced a classical dataset (Alg514) of 514 word problems, across various domains in algebra (such as percentages, mixtures, speeds, etc.). This dataset was annotated with multiple equations per problem. AddSub was introduced in Hosseini et al. (2014), with simple addition/subtraction problems, exhibiting limited language complexity. SingleOp (Roy et al., 2015) and MultiArith (Roy & Roth, 2015) were proposed such that there is a control over the operators (single operator in the former and two operators in the latter). SingleEq (Koncel-Kedziorski et al., 2015) is unique in incorporating long sentence structures for elementary level school problems. AllArith (Roy & Roth, 2017) is a subset of the union of AddSub, SingleEq and SingleOp. “Perturb” is a set of slightly perturbed word problems of AllArith, whereas Aggregate is the union of AllArith and Perturb. MAWPS (A **M**ath **W**ord **P**roblem **S**olving **R**epository) Koncel-Kedziorski et al. (2016) is a curated dataset (with deliberate template overlap control) that comprises all proposed datasets till that date. A single equation subset of MAWPS (AsDIV-A) Miao et al. (2020) has been studied, for diagnostic analysis of solvers. Similarly, the critique offered by Patel et al. (2021) was demonstrated using their newly proposed dataset SVAMP. In SVAMP, minutely perturbed word problems from the popular dataset AsDIV-A. This particular subset is used to demonstrate that, while high values of accuracy can be obtained on AsDIV-A easily, SVAMP poses a formidable challenge to most solvers, as it captures nuances in the relationship between similar language formation and dissimilar equations. All aforementioned datasets incorporate an annotation of both the *equation* and the *answer*. Given the subset-superset relationships between some of these datasets, empirical usage of these datasets would need to ensure careful sampling to creating subsets for training, testing and cross-validation.

## 5.2 | Large datasets

There are several types of datasets available for use. The following are some classifications the authors have observed.

### 5.2.1 | Algebraic large datasets

Dolphin18k Huang et al. (2016) is an early proprietary dataset that was evaluated primarily with the statistical solvers. GSM8k (Cobbe et al., 2021) is a recent single-equation dataset, that is the large scale version of AsDIV-A (Miao et al., 2020). Math23K is a popular Chinese dataset for single equation math word problem solving. A recent successor is Ape210k (Liang et al., 2021).

### 5.2.2 | Datasets with explanations

AQuA-RAT (Ling et al., 2017) introduced the first large crowd-sourced dataset for word problems with *rationales* or *explanations*. This makes the setting quite different from the aforementioned datasets, not only with respect to size, but also in the wide variety of domain areas (spanning physics, algebra, geometry, probability, etc.). Another point of

difference is that the annotation involves the entire textual explanation, rather than just the equations. MathQA (Amini et al., 2019) critically analyzed AQUA-RAT and selected the core subset and annotated it with a predicate list, to widen the remit of its usage. Once again, researchers must be mindful of the fact that MathQA is a subset of AQUA-RAT. MATH (Hendrycks et al., 2021) is another recent large dataset with 12,500 problems that encompasses a large set of domains and is currently one of the toughest word problems to solve.

## 6 | EVALUATION MEASURES

The most popular metric is *answer accuracy*, which evaluates the predicted equation and checks whether it is the same as the labeled one. The other metric is *equation accuracy*, which predominantly does string matching and assesses the match between the produced equation and the equation from the annotation label.

### 6.1 | Accuracy based measures

There are many ways of ascertaining the accuracy of the mathematical question answering. The most common ones utilized in algorithms were described previously—namely equation accuracy and answer accuracy. Given below is a more comprehensive list.

- *Answer accuracy*: Answer Accuracy measures the accuracy of the model in predicting the final mathematical answer.
- *Equation accuracy*: Equation Accuracy examines the extent of equation match. This may or may not test full equation equivalences.
- *Exact match accuracy*: Exact Match Accuracy measures the percentage of math word problems for which the model provides a solution that exactly matches the reference solution. This metric is useful for evaluating the precision of the model's output.
- *Variable match accuracy*: Variable Match Accuracy assesses the accuracy of the model in correctly identifying and utilizing the variables present in the math word problems. It provides insights into the model's understanding of the problem structure and its ability to handle varying input scenarios.
- *Expression tree depth accuracy*: Expression Tree Depth Accuracy evaluates the correctness of the hierarchical structure of the mathematical expressions generated by the model. It measures how well the model captures the depth of the expression tree compared to the ground truth. This is especially useful for graph-based solvers.

### 6.2 | Text generation evaluation

In the case of evaluating rationales for mathematical reasoning, it is possible to evaluate the quality of generated text using other metrics. These metrics are listed below.

- *BLEU score*: BLEU (Bilingual Evaluation Understudy) Score is a metric commonly used in natural language processing tasks, including text generation. It measures the similarity between the model's output and the reference solution based on n-gram precision. A higher BLEU score indicates better alignment with the reference.
- *ROUGE score*: ROUGE (Recall-Oriented Understudy for Gisting Evaluation) Score evaluates the quality of the model's output by comparing it to the reference solution in terms of overlapping n-grams. It is commonly used in text summarization tasks and provides a comprehensive assessment of content overlap.

### 6.3 | Human evaluation

With the advent of LLMs towards addressing MWPs, it is becoming imperative to develop a method of looking at logical fallacies in the text generated by LLMs. A human evaluation, though often expensive, is often the best arbiter available for a comprehensive evaluation.

## 7 | PERFORMANCE OF DEEP MODELS

In this section, we describe the performance of neural solvers towards providing the reader with a high-level view of the comparative performance across the several proposed models.

We have listed the performance of the deep models in Table 6, on two major datasets—Math23K and MAWPS. Some of these deep models report scores on other datasets as well. For conciseness, we have chosen the most popular datasets for deep models. We see that, in general, the models achieve around 70%–80% points on *answer accuracy*. Shen et al. (2021) outperforms all other models on Math23k whereas RPKHS (Yu et al., 2021) is the best model for MAWPS till date. As mentioned before, graph based models are both popular and effective. The pure LLMs are competitive with other neural solvers but the solvers that build on top of LLMs emerge as the formidable choice for MWP solvers.

A note of caution is that, as inferred from the discussion on datasets, (a) both Math23k and MAWPS are single equation datasets and (b) though some lexical overlap has been performed in the design of these two datasets, the semantic quality of these datasets are quite similar. This aspect has also been experimented and explored in Patel et al. (2021). Hence, though we present the best performing algorithms in this table, more research is required to design a suitable metric or a suitable dataset, such that one can conclusively compare these various algorithms. As a partial solution, new datasets the GSM8k and MATH are emerging as the de-facto standard for LLM-based MWP solving.

### 7.1 | Multi-domain datasets

Apart from these algebraic datasets, multi-domain datasets MathQA and AquA are also of special interest. This is described in Table 7 and depicted in Figure 5. The interesting takeaway is that, the addition of BERT modeling to AQUA (Piękos et al., 2021), still performed slightly worse than the Seq2Prog (Amini et al., 2019) model, which is a derivative of the Seq2Seq paradigm.

### 7.2 | Analysis and interpretation

By looking at the values, one can immediately notice that graph-based architectures possess significant advantages and are popular among model developers because they allow the developer to feed structural know-how into the model. The performance of each type of neural architecture is illustrated in Table 8 and Figure 6. While graph-based algorithms certainly perform well, they are eclipsed by the LLM based methods. The mammoth sizes of the models coupled with their modeling techniques make the gains larger than those obtained by graph-based models. The interesting thing to note is that graph-based models are actually performing better than GPT-3 and are competitive with many LLMs. Finally, the systems built on top of LLMs, being mindful of the domain, are the best performing. This suggests that domain knowledge injection is a key aspect of exploiting LLMs for MWP solving.

In Table 8 and Figure 6, the authors consolidate the performance across different types of models. Clearly, LLMs perform the best. However, the toughest dataset predicted till date is the MATH dataset (Table 9), where the best accuracy is a little over 50%. In the coming sections, the strengths and weaknesses of such LLMs for this task is described.

## 8 | ANALYSIS

In this chapter, we consider different aspects of studying MWP models including model design, dataset design, evaluation, how they come together and deliberate on possible future paths for the field of MWP solving, and by extension, knowledge-aware NLP.

### 8.1 | Analysis of neural solvers

In this section of the paper, we analyze the pros and cons of applying deep learning techniques to solve word problems automatically. At the outset, two layers of understanding are imperative: (i) linguistic structures that describe a situation or a sequence of events and (ii) mathematical structures that govern these language descriptions. Though deep

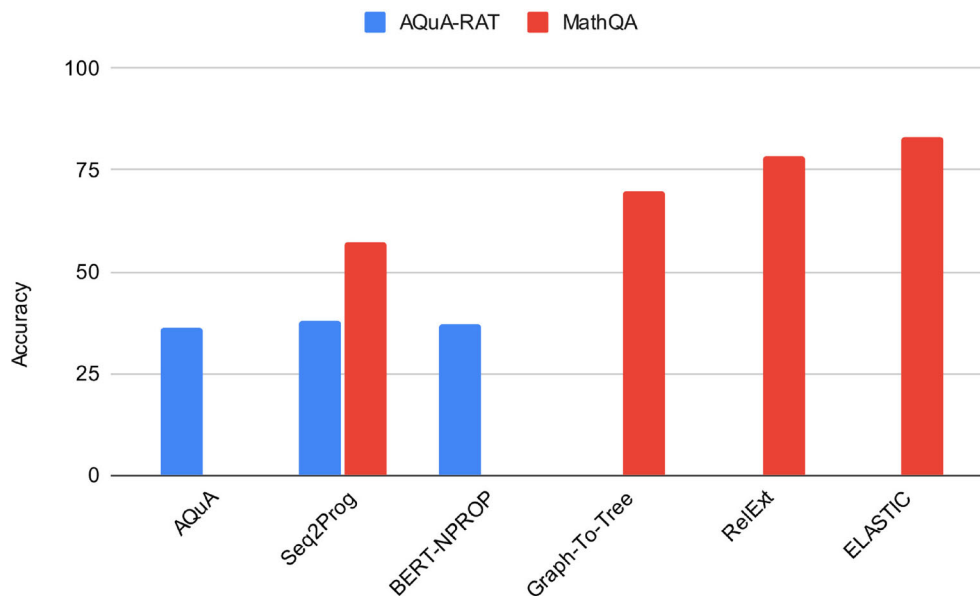


FIGURE 5 Performance on multi-domain datasets.

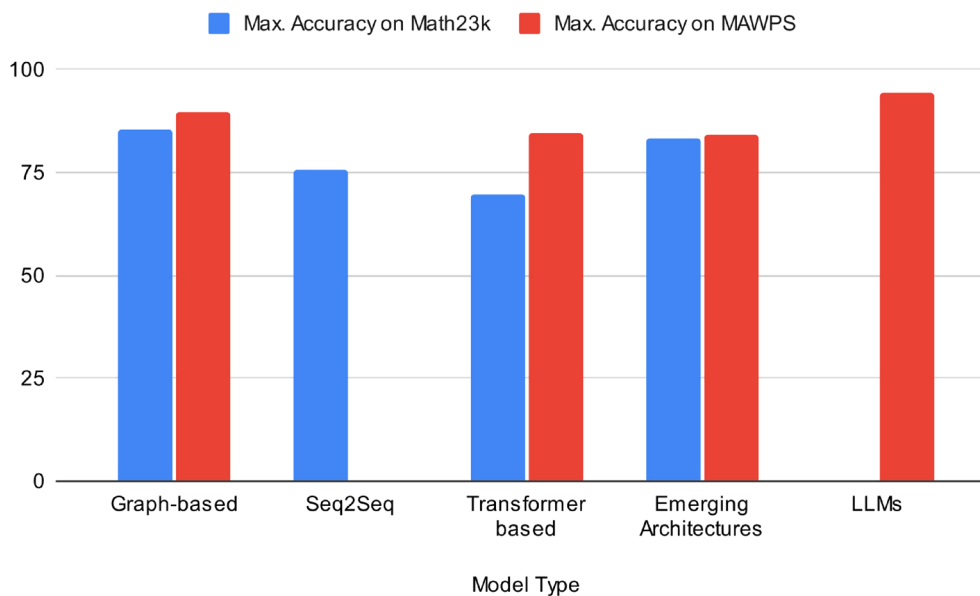


FIGURE 6 Overall accuracy.

learning models have rapidly scaled and demonstrated commendable results for capturing these two characteristics, a closer look reveals much potential for further exploration. The predominant modus-operandus is to create a deep model that converts the input natural language to the underlying equation. In some cases, the input is converted into a set of predicates (Amini et al., 2019) or explanations (Ling et al., 2017). The LLMs disrupted this design principle and latest gains in prompt tuning and instruction tuning, brought higher gains in accuracy, and more importantly, in explainability.

## 8.2 | What shortcuts are being learned?

Shortcut Learning (Geirhos et al., 2020) is a recently well-studied phenomenon of deep neural networks. It describes how deep learning models learn patterns in a shallow way and fall prey to questionable generalizations across datasets

TABLE 9 Answer accuracy of LLM based models on a hard dataset—MATH.

| Model       | Accuracy on MATH | Source                  |
|-------------|------------------|-------------------------|
| GPT-4       | 42.5             | OpenAI (2023)           |
| PaLM 2      | 42.8             | Anil et al. (2023)      |
| Minerva     | 50.3             | Lewkowycz et al. (2022) |
| PHP (GPT-4) | <b>54.3</b>      | Zheng et al. (2023)     |

Note: The best values are bolded.

(an example is an image being classified as sheep if there was grass alone; due to peculiarities in the dataset). In the context of word problems, Patel et al. (2021) exposed how removing the question and simply passing the situational context, leads to the correct equation being predicted. This suggests two things, issues with model design as well as issues with dataset design. The datasets have high equation template overlap, as well as text overlap. Word problem solving is a hard because two otherwise identical word problems, with a small word change (say changing the word *give* to *take*), would completely change the equation. Hence high lexical similarity does not translate to corresponding similarity in the mathematical realm (Patel et al., 2021; Sundaram et al., 2020), and attention to key aspects within the text is critical.

### 8.3 | Is accuracy enough?

As suggested by the discussion above, a natural line of investigation is to examine the evaluation measures, and perhaps the error measures for the deep models, in order to bring about a closer coupling between syntax and semantics. High accuracy of the models to predicting the answer or the equation suggests a shallow mapping between the text and the mathematical symbols. This is analogous to the famously observed McNamara fallacy,<sup>1</sup> which cautions against the overuse of a single metric to evaluate a complex problem. One direction of exploration is data augmentation with a single word problem annotated with multiple equivalent equations. Metrics that measure the soundness of the equations generated, the robustness of the model to simple perturbations (perhaps achieved using a denoising autoencoder) and the ability of the model to discern important entities in a word problem (perhaps using an attention analysis based metric), are the need of the future. An endeavor has been done by Kumar et al. (2021), where adversarial examples have been generated and utilized to evaluate SOTA models.

### 8.4 | Are the trained models accessible?

Most of the SOTA systems come with their own, well-documented repositories. Though an aggregated toolkit (Lan et al., 2021) (open-source MIT License) is available, running saved models in inference mode, to probe the quality of the datasets, proved to be a hard task, with varying missing hyper-parameters or missing saved models. This, however, interestingly suggests that API's that can take a single word problem as input and computes the output, would be highly useful for application designers. This has been done in the earlier systems such as Roy and Roth (2018) and Wolfram (2015).

### 8.5 | Is numeracy being understood?

Studies (Spithourakis & Riedel, 2018; Wallace et al., 2019) have delved deep into the numerical and mathematical ability of language models. The consensus is that, as a pattern recognition and language model, numerical common sense is hard to capture. The primary roadblock is that numbers are infinite and the vectorial representations of words seem to differ markedly from the semantics of the continuous space of numerals. A plethora of techniques have proposed to overcome this limitation (Duan et al., 2021; Geva et al., 2020; Jiang et al., 2019; Petrak et al., 2023; etc.). The common theme is be mindful of the challenges of numeracy and adding appropriate training objectives or methodologies such as contrastive learning. NumGLUE (Mishra et al., 2022) is a set of benchmarks that evaluates numeracy in different ways. Another approach of improving accuracy is to add widgets as GPT-4 (OpenAI, 2023) has done, which can ensure

mathematical accuracy. Yet another approach is to use program-aided language models (Gao et al., 2023) where the programming models can be used to extract meaningful numeric information. We have noticed that in general- GPT4 makes less arithmetic errors during language generation.

## 8.6 | Are semantics between language and math captured?

The authors would like to use an example (Figure 7) to drive home this point.

Given an erroneous word problem, where GPT is posed with a word problem that cannot be solved, GPT 3.5 and GPT 4 attempt to provide an answer. The maximum sum of two digits is that of 99, that is 18. GPT 3.5 postulated the answer to be 198. While the sum of digits is 20, it is a three digit number. GPT-4 fares better by presenting a two digit number 82. However, the sum of the digits is 10, not 20. This is one of many examples the authors have examined. Any proposition of expectation failure, in the format of word problems, provides interesting failure cases for further systematic study. This trend suggests further research is required, albeit having superior answer accuracy, language models still have major lapses in their conceptual clarity.

## 9 | ANALYSIS OF BENCHMARK DATASETS

In this section of the paper, we explore the various dimensions of the popular datasets (Table 6) with a critical and constructive perspective.

### 9.1 | Low resource setting

Compared to usual text related tasks, the available datasets are quite small in size. They also suffer from a large lexical overlap (Amini et al., 2019). This taxes algorithms, that now have to generalize from an effectively small dataset. The fact that the field of word problem solving is niche, where we cannot simply lift text from generic sources like Wikipedia, is one of the primary reasons why these datasets are small. Language precision is required, while maintaining mathematical sense. Hence, language generation is also a hard task.

|          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Question | A two digit number is twice its reverse. The sum of the digits is 20. Find the number.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| GPT-3.5  | <p>Let's represent the two-digit number as <math>10a + b</math>, where <math>a</math> and <math>b</math> are the digits in the tens and ones places, respectively. The reverse of the number is <math>10b + a</math>. The problem tells us that the number is twice its reverse, so we can write an equation:</p> $10a + b = 2(10b + a)$ <p>Simplifying and solving for <math>a</math> in terms of <math>b</math>, we get:</p> $10a + b = 20b + 2a \implies 8a = 19b \implies a = \frac{19}{8}b$ <p>Since <math>a</math> and <math>b</math> are both integers, <math>b</math> must be a multiple of 8 and <math>a</math> must be a multiple of 19. The only multiple of 8 that has a digit sum of 20 is 80, and the only multiple of 19 that is a digit is 19. Therefore, <math>a = 19</math> and <math>b = 8</math>. So the two-digit number is <math>10a + b = 198</math>, and its reverse is <math>10b + a = 89</math>. We can check that 198 is indeed twice 89, and that the sum of its digits is <math>1 + 9 + 8 = 18</math>, which is consistent with the problem statement.</p> <p><b>Therefore, the number is 198.</b></p> |
| GPT-4    | <p>Let's call the tens digit of the number "<math>a</math>" and the ones digit "<math>b</math>". The number can be expressed as <math>10a + b</math>. We know that the sum of the digits is 20, so <math>a + b = 20/2 = 10</math>. We also know that the number is twice its reverse, so <math>10a + b = 2(10b + a)</math>. Simplifying this equation gives us <math>9a = 19b</math>. Since <math>a</math> and <math>b</math> are digits, we know that <math>a</math> must be less than or equal to 9 and greater than or equal to 1. Similarly, <math>b</math> must be less than or equal to 9 and greater than or equal to 0. The only solution that satisfies all of these conditions is <math>a = 8</math> and <math>b = 2</math>.</p> <p><b>Therefore, the number is 82.</b></p>                                                                                                                                                                                                                                                                                                                                               |

**FIGURE 7** Analyzing GPT on math word problems. On posing an incorrect word problem, GPT3.5 and GPT 4 provide incorrect reasoning steps.

## 9.2 | Annotation cost

The datasets currently have little to no annotation costs involved as they are usually scraped from homework websites. There are some exceptions that involve crowd-sourcing (Ling et al., 2017) or intermediate representations apart from equations (Amini et al., 2019).

## 9.3 | Template overlap

Many studies (Zhang, Wang, Zhang, et al., 2020) have demonstrated that there is a high lexical and mathematical overlap between the word problems in popular datasets. While lexical overlap is desirable in a principled fashion, as demonstrated by Patel et al. (2021), it often limits the diversity and thus utility of the datasets. Consequently, many strategies have been adopted to mitigate such issues. Early attempts include controlling linguistic and equation template overlap (Koncel-Kedziorski et al., 2016; Miao et al., 2020). Later ideas revolve around controlled design and quality control of crowd-sourcing (Amini et al., 2019).

## 10 | ROAD AHEAD

In this section, we describe exciting frontiers of research for word problem solving algorithms.

### 10.1 | Semantic parsing

As rightly suggested by Zhang, Wang, Zhang, et al. (2020), the closest natural language task for word problem solving is that of *semantic parsing*, and not *translation* as most of the deep learning models have modeled. The mapping between extremely long chunks of text to short equation sentences has the advantage of generalizing on the decoder side, but equally has the danger of overloading many involved semantics into a simplistic equation model. To illustrate, an equation may be derived after applying a sequence of steps that is lost in a simple translation process. A lot of efforts have already been employed in adding such nuances in the modeling. One way is to model the input intelligently (for e.g., Liang et al., 2021) Here, sophisticated embeddings are learned from LLMs based models, using the word problem text as a training bed. The intermediate representations include simple predicates (Roy & Roth, 2018), while others involve a programmatic description (Amini et al., 2019; Ling et al., 2017). Yet another way is to include semantic information in the form of graphs as shown in Chiang and Chen (2019), Huang et al. (2018), Li et al. (2020), Qin et al. (2020), etc.). The authors have previously described popular modeling methods employed by LLMs such as *prompt engineering*, *CoT modeling*, *adapter tuning* and *program aided language approaches* as possible candidates for future research.

### 10.2 | Informed dataset design

As most datasets are sourced from websites, there is bound to be repetition. Efforts invested in modeling things such as the following could help aiding word problem research: (a) different versions of the same problem, (b) different equivalent equation types, (c) semantics of the language and the math. A step in this direction has been explored by Patel et al. (2021), which provides a challenge dataset for evaluating word problems, and (Kumar et al., 2021) where adversarial examples are automatically generated.

#### 10.2.1 | Dataset augmentation

A natural extension of dataset design, is dataset augmentation. Augmentation is a natural choice when we have datasets that are small and focused on a single domain. Then, linguistic and mathematical augmentation can be automated by domain experts. While template overlap is a concern in dataset design, it can be leveraged in contrastive

designs as in Sundaram et al. (2020), Li, Zhang, et al. (2021). A principled approach of reversing operators and building equivalent expression trees for augmentation has been explored here (Liu et al., 2022).

### 10.2.2 | Few shot learning

This is useful if we have a large number of non-annotated word problems or if we can come up with complex annotations (that capture semantics) for a small set of word problems. In this way *few shot learning* can generalize from few annotated examples.

## 10.3 | Knowledge aware models

We propose that word problem solving is more involved than even semantic parsing. From an intuitive space, we learn language from examples and interactions but we need to be explicitly *trained* in math to solve word problems (Marshall, 1996). This suggests we need to include mathematical models into our deep learning models to build generalizability and robustness. As mentioned before, a common approach is to include domain knowledge as a graph (Chiang & Chen, 2019; Qin et al., 2020, 2021; Wu et al., 2020).

The existing datasets describe word problems and expect the models to learn complex mathematical concepts. The datasets used are nowhere near the size required for robust training. The most popular dataset is GSM 8k, which as the name suggests, has a mere 8000 word problems. The models race to better each other and this benchmark can be taken as solved. However, it does not prove that the LLMs have understood the underlying concepts of these word problems.

## 10.4 | Improvements on evaluation

The use of accuracy or equation matching could possibly hide some of the linguistic nuances of the word problem structure. This is indirectly highlighted by the abysmal performance of SOTA models on the challenge dataset proposed by Patel et al. (2021). Hence, some research directions are required to develop stronger metrics. One of the ways of doing that is to develop a benchmark for language models akin to Mishra et al. (2022). This particular benchmark is used to expose mathematical discrepancies in language models. Benchmark tasks and datasets can be an interesting alternative to metric design for such word problem solvers.

### 10.4.1 | Concept evaluation

Apart from the datasets on word problems, datasets are also required on concept question answering to reveal the extent of understanding by NLP systems, not merely answer accuracy.

### 10.4.2 | Confidence evaluation

LLMs currently provide mathematical answers with confidence, even erroneous ones. One possible metric to explore is in evaluation of the confidence in the case of both correct and wrong answers and evaluate these models in a weighted manner.

## 11 | CONCLUSION

In this paper, we surveyed the existing math word problem solvers, with a focus on deep learning models. Word problem solving is a case study for deep semantic parsing—a task that is often described as AI-complete. This survey traced the history of mathematical word problem solving right from the 1960s. We began with the symbolic solvers that

employed rule-based algorithms to match complex linguistic patterns to templates that encoded mathematical information. These algorithms tended to work in niche domains and comparison was often difficult, we then presented statistical solvers that utilized human engineering efforts to develop a vectorized representation of word problems and used machine learning techniques to model math word problem solving as a retrieval or classification problem. This approach introduced many datasets for comparable analysis. However, these methods performed better linguistic processing but limited mathematical modeling.

Deep models were initially majorly modeled as encoder-decoder models, with input as text and decoder output as equations. We listed several interesting formulations of this paradigm—namely Seq2Seq models, graph-based models, transformer-based models, emerging architectures and LLMs. Prior to LLM based methods, linguistic complexity was captured by sequence modeling in varying architectures such as LSTM, transformers and so on. Graph based methods and other unique architectures modeled complex domain knowledge relationships. In general, LLMs tend to capture complex structural elements that can benefit both linguistic and mathematical aspects. We then explored in detail the various datasets in use. Subsequently, we analyzed the various approaches of modeling word problem solving, followed by the characteristics of the popular datasets. We saw an overwhelming trend that paying heed to the mathematical modeling and tying to the linguistic aspects reaped rich dividends. We concluded that the brittleness of the SOTA models was due to: (a) modeling decisions, and (b) dataset design. This is intended as a comprehensive survey, but the authors acknowledge that there may be methods that have escaped their attention. We also caution that the analysis provided could be subjective and opinionated, and there could be legitimate disagreements with the perspectives put forward. Finally, we mentioned few avenues of further exploration such as the use of semantically rich models, informed dataset design and incorporation of domain knowledge.

## AUTHOR CONTRIBUTIONS

**Sowmya S. Sundaram:** Conceptualization (equal); writing – original draft (lead). **Sairam Gurajada:** Conceptualization (equal); formal analysis (equal); supervision (equal); writing – review and editing (equal). **Deepak Padmanabhan:** Conceptualization (equal); supervision (equal); writing – review and editing (equal). **Savitha Sam Abraham:** Conceptualization (equal); writing – review and editing (supporting). **Marco Fisichella:** Conceptualization (equal); supervision (equal); writing – review and editing (equal).

## ACKNOWLEDGMENT

Open Access funding enabled and organized by Projekt DEAL.

## CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

## DATA AVAILABILITY STATEMENT

Data sharing is not applicable as no new data was created or analysed in this study.

## ORCID

Sowmya S. Sundaram  <https://orcid.org/0000-0002-0086-7582>

Marco Fisichella  <https://orcid.org/0000-0002-6894-1101>

## RELATED WIREs ARTICLES

[Emerging directions in predictive text mining](#)

[Deep learning for sentiment analysis: A survey](#)

[Automatic question generation](#)

## ENDNOTE

<sup>1</sup> [https://en.wikipedia.org/wiki/McNamara\\_fallacy](https://en.wikipedia.org/wiki/McNamara_fallacy)

## REFERENCES

Amini, A., Gabriel, S., Lin, S., Koncel-Kedziorski, R., Choi, Y., & Hajishirzi, H. (2019). MathQA: Towards interpretable math word problem solving with operation-based formalisms. *Proceedings of the 2019 Conference of the North American Chapter of the Association for*

- Computational Linguistics: Human Language technologies, Vol. 1 (Long and Short Papers)*, 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.
- Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., Chu, E., Clark, J. H., Shafey, L. E., Huang, Y., Meier-Hellstern, K., Mishra, G., Moreira, E., Omernick, M., Robinson, K., ... Wu, Y. (2023). Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.
- Bakman, Y. (2007). Robust understanding of word problems with extraneous information. *arXiv preprint math/0701393*.
- Bobrow, D. G. (1964). A question-answering system for high school algebra word problems. *Proceedings of the October 27–29, 1964, fall joint computer conference, part I*, 591–614. ACM.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners.
- Cao, Y., Hong, F., Li, H., & Luo, P. (2021). A bottom-up dag structure extraction model for math word problems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(1), 39–46.
- Chen, J., Tang, J., Qin, J., Liang, X., Liu, L., Xing, E., & Lin, L. (2021). GeoQA: A geometric question answering benchmark towards multi-modal numerical reasoning. *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 513–523, Online. Association for Computational Linguistics.
- Chiang, T.-R., & Chen, Y.-N. (2019). Semantically-aligned equation generation for solving and reasoning math word problems. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1 (Long and Short Papers)*, 2656–2668, Minneapolis, Minnesota. Association for Computational Linguistics.
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734, Doha, Qatar. Association for Computational Linguistics.
- Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., ... Noah Fiedel, N. (2022). Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.
- Cobbe, K., Kosaraju, V., Bavarian, M., Hilton, J., Nakano, R., Hesse, C., & Schulman, J. (2021). Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.
- Dellarosa, D. (1986). A computer simulation of children's arithmetic word-problem solving. *Behavior Research Methods, Instruments, & Computers*, 18(2), 147–154.
- Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.
- Dries, A., Kimmig, A., Davis, J., Belle, V., & De Raedt, L. (2017). Solving probability problems in natural language. *Proceedings Twenty-Sixth International Joint Conference on Artificial Intelligence*, 3981–3987.
- Duan, H., Yang, Y., & Tam, K. Y. (2021). Learning numeracy: A simple yet effective number embedding approach using knowledge graph. *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2597–2602.
- Faldu, K., Sheth, A. P., Kikani, P., Gaur, M., & Avasthi, A. (2021). Towards tractable mathematical reasoning: Challenges, strategies, and opportunities for solving math word problems. *CoRR*, abs/2111.05364.
- Feng, W., Liu, B., Xu, D., Zheng, Q., & Xu, Y. (2021). GraphMR: Graph neural network for mathematical reasoning. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3395–3404, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Fletcher, C. R. (1985). Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers*, 17(5), 565–571.
- Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., & Neubig, G. (2023). Pal: Program-aided language models. *International Conference on Machine Learning*, pages 10764–10799. PMLR.
- Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11), 665–673.
- Geva, M., Gupta, A., & Berant, J. (2020). Injecting numerical reasoning skills into language models. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 946–958.
- Griffith, K., & Kalita, J. (2020). Solving arithmetic word problems using transformer and pre-processing of problem texts. *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, 76–84, Indian Institute of Technology Patna, Patna, India. NLP Association of India (NLP AI).
- He, R., Liu, L., Ye, H., Tan, Q., Ding, B., Cheng, L., Low, J., Bing, L., & Si, L. (2021). On the effectiveness of adapter-based tuning for pre-trained language model adaptation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Vol. 1: Long Papers)*, 2208–2222.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., & Steinhardt, J. (2021). Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2(4), 6.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.
- Hong, Y., Li, Q., Cio, D., Huang, S., & Zhu, S.-C. (2021). Learning by fixing: Solving math word problems with weak supervision. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):4959–4967.

- Hosseini, M. J., Hajishirzi, H., Etzioni, O., & Kushman, N. (2014). Learning to solve arithmetic word problems with verb categorization. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 523–533.
- Huang, D., Shi, S., Lin, C.-Y., & Yin, J. (2017). Learning fine-grained expressions to solve math word problems. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 805–814.
- Huang, D., Shi, S., Lin, C.-Y., Yin, J., & Ma, W.-Y. (2016). How well do computers solve math word problems? Large-scale dataset construction and evaluation. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers)*, 887–896, Berlin, Germany. Association for Computational Linguistics.
- Huang, D., Yao, J.-G., Lin, C.-Y., Zhou, Q., & Yin, J. (2018). Using intermediate representations to solve math word problems. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers)*, 419–428, Melbourne, Australia. Association for Computational Linguistics.
- Jiang, C., Nian, Z., Guo, K., Chu, S., Zhao, Y., Shen, L., & Tu, K. (2019). Learning numeral embeddings. *arXiv preprint arXiv:2001.00003*.
- Jie, Z., Li, J., & Lu, W. (2022). Learning to reason deductively: Math word problem solving as complex relation extraction. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers)*, 5944–5955, Dublin, Ireland. Association for Computational Linguistics.
- Kim, B., Ki, K. S., Lee, D., & Gweon, G. (2020). Point to the expression: Solving algebraic word problems using the expression-pointer transformer model. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 3768–3779, Online. Association for Computational Linguistics.
- Kim, B., Ki, K. S., Rhim, S., & Gweon, G. (2022). Ept-x: An expression-pointer transformer model that generates explanations for numbers. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers)*, 4442–4458.
- Koch, G., Zemel, R., & Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. *ICML Deep Learning Workshop, Vol. 2*.
- Koncel-Kedziorski, R., Hajishirzi, H., Sabharwal, A., Etzioni, O., & Ang, S. D. (2015). Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3, 585–597.
- Koncel-Kedziorski, R., Roy, S., Amini, A., Kushman, N., & Hajishirzi, H. (2016). Mawps: A math word problem repository. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1152–1157.
- Kumar, V., Maheshwary, R., & Pudi, V. (2021). Adversarial examples for evaluating math word problem solvers. *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2705–2712, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Kushman, N., Artzi, Y., Zettlemoyer, L., & Barzilay, R. (2014). Learning to automatically solve algebra word problems. *ACL Anthology*, 1, 271–281.
- Lan, Y., Wang, L., Zhang, Q., Lan, Y., Dai, B. T., Wang, Y., Zhang, D., & Lim, E.-P. (2021). Mwptoolkit: An open-source framework for deep learning-based math word problem solvers. *arXiv preprint arXiv:2109.00799*.
- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning* (pp. 1188–1196). PMLR.
- Le-Khac, P. H., Healy, G., & Smeaton, A. F. (2020). Contrastive representation learning: A framework and review. *IEEE Access*, 8, 193907–193934.
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.
- Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., & Wu, Y. (2022). Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35, 3843–3857.
- Li, J., Wang, L., Zhang, J., Wang, Y., Dai, B. T., & Zhang, D. (2019). Modeling intra-relation in math word problems with different functional multi-head attentions. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 6162–6167, Florence, Italy. Association for Computational Linguistics.
- Li, L., Lin, Y., Ren, S., Li, P., Zhou, J., & Sun, X. (2021). Dynamic knowledge distillation for pre-trained language models. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 379–389, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Li, S., Wu, L., Feng, S., Xu, F., Xu, F., & Zhong, S. (2020). Graph-to-tree neural networks for learning structured input-output translation with applications to semantic parsing and math word problem. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2841–2852, Online. Association for Computational Linguistics.
- Li, Z., Zhang, W., Yan, C., Zhou, Q., Li, C., Liu, H., & Cao, Y. (2021). Seeking patterns, not just memorizing procedures: Contrastive learning for solving math word problems. *CoRR*, abs/2110.08464.
- Liang, Z., Zhang, J., Shao, J., & Zhang, X. (2021). MWP-BERT: A strong baseline for math word problems. *CoRR*, abs/2107.13435.
- Liang, Z., & Zhang, X. (2021). Solving math word problems with teacher supervision. Zhou, Z.-H., (Ed.), *Proceedings of the thirtieth international joint conference on artificial intelligence, IJCAI-21*, 3522–3528. International Joint Conferences on Artificial Intelligence Organization. Main Track.

- Lin, X., Huang, Z., Zhao, H., Chen, E., Liu, Q., Wang, H., & Wang, S. (2021). Hms: A hierarchical solver with dependency-enhanced understanding for math word problem. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5), 4232–4240.
- Ling, W., Yogatama, D., Dyer, C., & Blunsom, P. (2017). Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.
- Liu, Q., Guan, W., Li, S., Cheng, F., Kawahara, D., & Kurohashi, S. (2022). Roda: Reverse operation based data augmentation for solving math word problems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30, 1–11.
- Liu, Q., Guan, W., Li, S., & Kawahara, D. (2019). Tree-structured decoding for solving math word problems. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2370–2379, Hong Kong, China. Association for Computational Linguistics.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Marshall, S. P. (1996). *Schemas in problem solving*. Cambridge University Press.
- Miao, S.-Y., Liang, C.-C., & Su, K.-Y. (2020). A diverse corpus for evaluating and developing English math word problem solvers. *Proceedings of the 58th annual meeting of the Association for Computational Linguistics*, 975–984, Online. Association for Computational Linguistics.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119). MIT Press.
- Mishra, S., Mitra, A., Varshney, N., Sachdeva, B., Clark, P., Baral, C., & Kalyan, A. (2022). NumGLUE: A suite of fundamental yet challenging mathematical reasoning tasks. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long papers)*, 3505–3523, Dublin, Ireland. Association for Computational Linguistics.
- Mitra, A., & Baral, C. (2016). Learning to use formulas to solve simple arithmetic problems. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers)*, 2144–2153, Berlin, Germany. Association for Computational Linguistics.
- Mukherjee, A., & Garain, U. (2008). A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2), 93–122.
- OpenAI. (2023). Gpt-4 technical report.
- Patel, A., Bhattamishra, S., & Goyal, N. (2021). Are NLP models really able to solve simple math word problems? *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2080–2094, Online. Association for Computational Linguistics.
- Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *Proc. of NAACL*.
- Petrak, D., Moosavi, N. S., & Gurevych, I. (2023). Arithmetic-based pretraining improving numeracy of pretrained language models. *Proceedings of the 12th joint Conference on Lexical and Computational Semantics (\*SEM 2023)*, 477–493.
- Piękos, P., Malinowski, M., & Michalewski, H. (2021). Measuring and improving BERT's mathematical abilities by predicting the order of reasoning. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Vol. 2: Short Papers)*, 383–394, Online. Association for Computational Linguistics.
- Qin, J., Liang, X., Hong, Y., Tang, J., & Lin, L. (2021). Neural-symbolic solver for math word problems with auxiliary tasks. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Vol. 1: Long Papers)*, 5870–5881, Online. Association for Computational Linguistics.
- Qin, J., Lin, L., Liang, X., Zhang, R., & Lin, L. (2020). Semantically-aligned universal tree-structured solver for math word problems. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 3780–3789, Online. Association for Computational Linguistics.
- Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1), 5485–5551.
- Roy, S., & Roth, D. (2015). Solving general arithmetic word problems. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.
- Roy, S., & Roth, D. (2017). Unit dependency graph and its application to arithmetic word problem solving. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI'17*, 3082–3088. AAAI Press.
- Roy, S., & Roth, D. (2018). Mapping to declarative knowledge for word problem solving. *Transactions of the Association of Computational Linguistics*, 6, 159–172.
- Roy, S., Vieira, T., & Roth, D. (2015). Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3, 1–13.
- Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., Biderman, S., Webson, A., Ammanamanchi, P. S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A. V., ... Wolf, T. (2022). Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.
- Schick, T., Dwivedi-Yu, J., Dess, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

- Seo, M., Hajishirzi, H., Farhadi, A., Etzioni, O., & Malcolm, C. (2015). Solving geometry problems: Combining text and diagram interpretation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1466–1476.
- Shen, J., Yin, Y., Li, L., Shang, L., Jiang, X., Zhang, M., & Liu, Q. (2021). Generate & rank: A multi-task framework for math word problems. *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2269–2279, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Shen, Y., & Jin, C. (2020). Solving math word problems with multi-encoders and multi-decoders. *Proceedings of the 28th International Conference on Computational Linguistics*, 2924–2934, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Shi, S., Wang, Y., Lin, C., Liu, X., & Rui, Y. (2015). Automatically solving number word problems by semantic parsing and reasoning. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, September 17-21, 2015*, 1132–1142.
- Spithourakis, G., & Riedel, S. (2018). Numeracy for language models: Evaluating and improving their ability to predict numbers. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers)*, 2104–2115.
- Sundaram, S. S., & Abraham, S. S. (2019). Semantic representation for age word problems with schemas. *New Generation Computing*, 37(4), 429–452.
- Sundaram, S. S., Deepak, P., & Abraham, S. S. (2020). Distributed representations for arithmetic word problems. *Thirty-Fourth AAAI Conference on Artificial Intelligence, Distributed Representations for Arithmetic Word Problems*.
- Suster, S., Fivez, P., Totis, P., Kimmig, A., Davis, J., de Raedt, L., & Daelemans, W. (2021). Mapping probability word problems to executable representations. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3627–3640, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Proceedings of the 27th International Conference on Neural Information Processing Systems – Vol. 2, NIPS'14*, 3104–3112, Cambridge, MA, USA. MIT Press.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
- Tsai, S.-H., Liang, C.-C., Wang, H.-M., & Su, K.-Y. (2021). Sequence to general tree: Knowledge-guided geometry word problem solving. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Vol. 2: Short Papers)*, 964–972, Online. Association for Computational Linguistics.
- Upadhyay, S., & Chang, M.-W. (2017). Annotating derivations: A new evaluation strategy and dataset for algebra word problems. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Vol. 1, Long Papers*, 494–504, Valencia, Spain. Association for Computational Linguistics.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. U., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems, Vol. 30* Curran Associates, Inc. MIT Press.
- Wallace, E., Wang, Y., Li, S., Singh, S., & Gardner, M. (2019). Do nlp models know numbers? Probing numeracy in embeddings. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 5307–5315.
- Wang, L., Zhang, D., Gao, L., Song, J., Guo, L., & Shen, H. T. (2018). Mathdqn: Solving arithmetic word problems via deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 5545–5552.
- Wang, T., & Lu, W. (2023). Learning multi-step reasoning by solving arithmetic tasks. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Vol. 2: Short Papers)*, 1229–1238.
- Wang, Y., Liu, X., & Shi, S. (2017). Deep neural solver for math word problems. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 845–854.
- Wolfram, S. (2015). Wolfram|alpha. On the WWW. URL <http://www.wolframalpha.com>
- Wu, Q., Zhang, Q., Fu, J., & Huang, X. (2020). A knowledge-aware sequence-to-tree network for math word problem solving. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 7137–7146, Online. Association for Computational Linguistics.
- Wu, Q., Zhang, Q., & Wei, Z. (2021). An edge-enhanced hierarchical graph-to-tree network for math word problem solving. *Findings of the Association for Computational Linguistics: EMNLP 2021*, 1473–1482, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Wu, Q., Zhang, Q., Wei, Z., & Huang, X. (2021). Math word problem solving with explicit numerical values. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing. Vol. 1: Long Papers*, 5859–5869, Online. Association for Computational Linguistics.
- Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1), 4–24.
- Xia, M., Huang, G., Liu, L., & Shi, S. (2019). Graph based translation memory for neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33*, 7297–7304.
- Xie, Y., Kawaguchi, K., Zhao, Y., Zhao, X., Kan, M.-Y., He, J., & Xie, Q. (2023). Decomposition enhances reasoning via self-evaluation guided decoding. *arXiv preprint arXiv:2305.00633*.
- Xie, Z., & Sun, S. (2019). A goal-driven tree-structured neural model for math word problems. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 5299–5305. International Joint Conferences on Artificial Intelligence Organization.

- Yu, W., Wen, Y., Zheng, F., & Xiao, N. (2021). Improving math word problems with pre-trained knowledge and hierarchical reasoning. *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3384–3394, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2020). Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33, 17283–17297.
- Zaporozhets, K., Bekoulis, G., Deleu, J., Demeester, T., & Develder, C. (2021). Solving arithmetic word problems by scoring equations with recursive neural networks. *Expert Systems with Applications*, 174, 114704.
- Zhang, D., Wang, L., Zhang, L., Dai, B. T., & Shen, H. T. (2020). 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.
- Zhang, J., Lee, R. K.-W., Lim, E.-P., Qin, W., Wang, L., Shao, J., & Sun, Q. (2020). Teacher-student networks with multiple decoders for solving math word problem. Bessiere, C., (Ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20* (pp. 4011–4017). International Joint Conferences on Artificial Intelligence Organization. Main track.
- Zhang, J., & Moshfeghi, Y. (2022). Elastic: Numerical reasoning with adaptive symbolic compiler. *Advances in Neural Information Processing Systems*, 35(2022), 12647–12661.
- Zhang, J., Wang, L., Lee, R. K.-W., Bin, Y., Wang, Y., Shao, J., & Lim, E.-P. (2020). Graph-to-tree learning for solving math word problems. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 3928–3937, Online. Association for Computational Linguistics.
- Zhao, X., Xie, Y., Kawaguchi, K., He, J., & Xie, Q. (2023). Automatic model selection with large language models for reasoning. *arXiv preprint arXiv:2305.14333*.
- Zheng, C., Liu, Z., Xie, E., Li, Z., & Li, Y. (2023). Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv:2304.09797*.
- Zhou, L., Dai, S., & Chen, L. (2015). Learn to solve algebra word problems using quadratic programming. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 817–822, Lisbon, Portugal. Association for Computational Linguistics.

**How to cite this article:** Sundaram, S. S., Gurajada, S., Padmanabhan, D., Abraham, S. S., & Fisichella, M. (2024). Does a language model “understand” high school math? A survey of deep learning based word problem solvers. *WIREs Data Mining and Knowledge Discovery*, 14(4), e1534. <https://doi.org/10.1002/widm.1534>