Turkish Journal of Electrical Engineering and Computer Sciences

Volume 31 | Number 2 Article 13

1-1-2023

Solving Turkish math word problems by sequence-to-sequence encoder-decoder models

ESİN GEDİK

TUNGA GÜNGÖR

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2023) 31: 431 – 447
© TÜBİTAK
doi:10.55730/1300-0632.3993

Solving Turkish math word problems by sequence-to-sequence encoder-decoder models

Esin GEDİK*, **Tunga GÜNGÖR**
Department of Computer Engineering, Boğaziçi University, İstanbul, Turkey

Received: 24.10.2022 • **Accepted/Published Online:** 13.02.2023 • **Final Version:** 23.03.2023

Abstract: Solving math word problems (MWP) is a challenging task due to the semantic gap between natural language texts and mathematical equations. The main purpose of the task is to take a written math problem as input and produce a proper equation as output for solving that problem. This paper describes a sequence-to-sequence (seq2seq) neural model for automatically solving Turkish MWPs based on their semantic meanings in the text. It comprises a bidirectional encoder to comprehend the semantics of the problem by encoding the input sequence and a decoder with attention to extract the equation by tracking the semantic meanings of the output symbols. We investigate the success of several embedding types, pretrained language models, and neural models. Our research is novel in the sense that there exist no studies in Turkish on this natural language processing task that utilizes pretrained language models and neural models. There is also no Turkish dataset that can be used to train the neural models for the MWP task. As the first large-scale Turkish MWP dataset, we translated the well-known English MWP datasets into Turkish using a machine translation system. Although Turkish is an agglutinative and grammatically challenging language, the proposed models achieve around 72% accuracy on the dataset compiled from three English datasets.

Key words: Math word problems, sequence-to-sequence model, attention mechanism, natural language processing

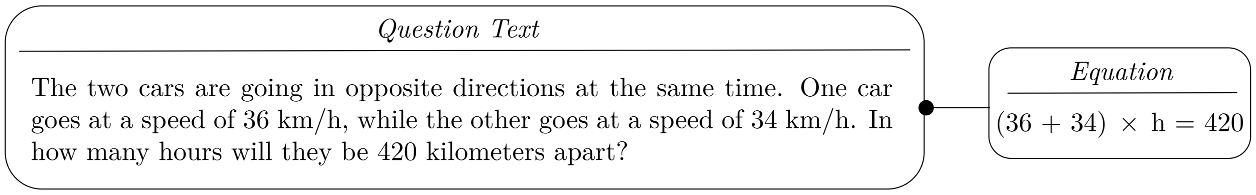
1. Introduction

A math word problem (MWP) is a descriptive paragraph that depicts a real-world event or scenario with mathematical relations. Figure 1 represents an MWP and its generated equation. A semantic parsing component receives a question text as input to create a mathematical expression. The output is an equation that symbolically describes the same mathematical relationship as in the example "(36 + 34) × h = 420". The structure of the equation may vary according to the designed system, but the ultimate goal is to find the numerical answer of the problem. In other words, an answer is sought for the variable "h" in the equation.

Generating an automatic answer to an MWP has not shown sufficient progress. One reason is the difficulty of mapping natural language text into machine-readable form in terms of the semantic meaning of each number in the problem. Another reason is that MWP complexity is determined by several dimensions including reasoning, linguistic complexity, and domain knowledge. For the last few decades, studies on solving MWPs using rule-based models have been conducted with a limited amount of data [1, 2]. These models require predefined problem and equation templates [3]. It is not possible to address problems that have never been encountered before or whose rules have not been stated. Because of this limitation, the MWP task has been adapted to be solved with the state-of-the-art neural models which provided further progress. There are many studies in the literature, particularly with English and Chinese MWP datasets comprised of manually

*Correspondence: esin.gedik@boun.edu.tr

constructed questions. However, there is neither a neural-based study nor a corpus for solving Turkish MWPs, and the number of rule-based research in Turkish is also relatively low [4–7].



Question Text

The two cars are going in opposite directions at the same time. One car goes at a speed of 36 km/h, while the other goes at a speed of 34 km/h. In how many hours will they be 420 kilometers apart?

Equation

$(36 + 34) \times h = 420$

Figure 1. An MWP example.

Based on the lack of MWP studies in Turkish, in this paper, we propose a deep neural solver to automatically solve Turkish MWPs. In contrast to prior rule-based techniques, we utilize a sequence-to-sequence (seq2seq) model with an attention mechanism to directly translate MWPs to equation templates. Seq2seq is a recurrent neural network (RNN) model, which employs an encoder to map the input sequence to a fixed-dimensional vector, followed by a decoder to generate the target sequence from this vector [8]. In our paper, the encoder is intended to comprehend the semantics of the question texts. In contrast, the decoder employs self-attention to vectorize the quantities and determine which symbol to generate at each time step. We use two types of seq2seq models as the long short-term memory (LSTM) encoder-decoder and the gated recurrent unit (GRU) encoder-decoder.

In neural models, word embeddings are used to represent words as real-valued vectors in a vector space. In this work, we employ the embedding algorithms which are word2vec, GloVe, and fastText, and pretrained language models which are BERT, ConvBERT, and ELECTRA to generate embeddings and compare them with each other. As a result of the comparative analysis, 72% accuracy and 73% bilingual evaluation understudy (BLEU) score are obtained with the GRU encoder-decoder model that uses BERT embeddings.

We also introduce new Turkish MWP corpora by translating and combining English benchmark datasets to solve elementary-level problems. We performed comprehensive manual arrangements and processing on the translated datasets. We publish the code of the models and the corpora consisting of question texts, equations, and answers for further research on Turkish MWP [1].

The contributions of the paper are as follows: (1) constructing the first Turkish MWP corpora to be used as benchmark datasets, (2) implementing the first neural-based models to solve Turkish MWP tasks, and (3) achieving high accuracies and thus creating a benchmark study for future studies on this task.

The rest of the paper is organized as follows: Section 2 presents the literature review on MWP. Section 3 introduces the novel Turkish MWP corpora built in this work. Section 4 describes the neural-based Turkish MWP solving models and methodologies. Section 5 provides the dataset statistics, machine translation evaluation, results of the experiments, comparison with other MWP studies, model parameters, and implementation details. Section 6 concludes the paper.

2. Related work

Previous studies on automatically solving MWPs can be grouped into three main categories in terms of the models used: rule-based models, statistical models, and neural models. Below we give a brief review of these studies and also cover Turkish MWP studies.

[1]https://github.com/esingedik/Turkish-MWP-Corpora-and-Code [accessed on September 2022]

2.1. Rule-based models

Most of the previous studies on solving MWPs focus on rule-based methods and feature engineering which requires expert knowledge on the domain [2]. Since these features are mostly at the lexical level, such models cannot actually capture the semantics of math problems. Bakman [3] solves free-format multistep MWPs by developing a schema-based system and defining several schemas such as creation and termination. After instantiating the schemas using lexical verb-based rules, the system forms the suitable equation. As an alternative approach, Liguda and Pfeiffer [9] suggest augmented semantic networks. In the network, nodes indicate quantities while edges indicate transition states. For multistep addition and subtraction MWPs, Yuhui et al. [10] represent the question text with frames to store the essential semantic relations.

2.2. Statistical models

Early statistical studies use semantic parsing methods [11] or machine learning methods [12–14] for solving MWPs. Semantic parsing methods attempt to establish a direct mapping from the question text to the equation structure. One of the well-known studies that uses this approach binds the specific lexicon-based properties to the equation operators [11]. The equation is then constructed by applying state transitions based on the operators activated by the verb categories. As a different approach, machine learning methods utilize classical learning models to recognize entities, quantities, and operators in the question text and provide the solution using a basic logical inference. These models improve the semantic parsing methods by predefined logic templates. Kushman et al. [12] extract templates about math expressions from the training data. Mitra and Baral [13] categorize addition and subtraction problems and propose problem templates. Upadhyay et al. [14] use a semisupervised technique to forecast templates and match alignments.

2.3. Neural models

As in many areas of natural language processing (NLP), neural models are frequently used in the solving of MWPs and successful results are obtained. Wang et al. [15] develop an RNN model to map the problem statements to the equation templates. They solve MWPs by generating the equation templates through a seq2seq model. Robaidek et al. [16] obtain equation templates using seq2seq models with attention mechanism and evaluate the model with both LSTM and convolutional neural network (CNN) as the encoder and the decoder. In another work [17], an intermediate meaning representation reduces the semantic gap between natural language and equations using attention and copy mechanism. As a different neural approach that improves the seq2seq model with reinforcement learning, the copy and alignment mechanism to the sequence-to-sequence model (CASS) is proposed [18]. The copy model directly copies the numbers in the question text to the equation, while the alignment model connects the numbers in the equations and the problem description.

2.4. Related studies on Turkish MWP

Although there are no studies based on neural models for solving Turkish MWP, there are a few studies that use rule-based and shallow models. The two earliest studies propose a method and build a system that can solve primary school-level MWPs [4, 5]. The system consists of several tasks which are morphological and semantic analysis of the words in the input, syntactic analysis of the sentences, and providing a description of the commonsense knowledge to allow correct answers to the problems. The basic idea is to have a template that maps sentences to semantic categories. Çakıroğlu [6] presents semantic networks and focuses on morphological, syntactic, and semantics analysis. Another study investigates morphological analysis of the words and definition of the problem types [7]. In order to determine which pattern the problem belongs to, the question sentence is divided into words and numbers, and it is decided whether it is addition, subtraction, multiplication, or division.

3. Datasets

The use of neural models in the MWP task necessitates datasets containing large amounts of data and equations. The simplest MWPs have a question text and a mathematical equation that includes numbers with basic arithmetic operators (+, -, /, $\times$). More complex MWPs have systems of equations, include other operators like square root and exponentiation, and contain specialized information such as volume calculation. Recent studies focus on solving complex and challenging MWPs, such as geometry problems [19, 20], multiple unknown linear problems [21], and IQ problems [22].

3.1. Available datasets

Studies based on English and Chinese datasets are pioneering studies in the MWP domain. Table 1 lists the benchmark datasets used in most of the studies. Dolphin18K is a large-scale and challenging dataset with 18,460 problems and 5871 equation templates [21]. Since the dataset is created from online forums, there may be errors in the equations and answers. MAWPS is a frequently used English benchmark dataset containing equation templates and 3320 questions [23]. In a research [24], a simplified version with only 1920 questions is used. ASDiv-A [25] is a diverse dataset in terms of lexicon patterns and problem types. This dataset also includes additional domain knowledge. SVAMP is a challenging and newly introduced dataset by Patel et al. [24] and includes several types of modifications to a set of seed problems derived from the ASDiv-A dataset. Each equation template is converted to prefix form and all numbers are masked with a metasymbol. MathQA is a massive dataset of 37,200 English multiple-choice MWPs in geometry, physics, and probability domains [26]. Math23K, one of the most well-known Chinese MWP datasets, is composed of 23,161 questions [15].

Table 1. Statistics of MWP datasets.

Dataset	# Problems	Operators	Problem types	Language
Dolphin18K	18460	+, -, /, $\times$	linear, nonlinear	English
MAWPS	1920 or 3320	+, -, /, $\times$	algebra	English
ASDiv-A	1218	+, -, /, $\times$	algebra, linear	English
SVAMP	1000	+, -, /, $\times$	algebra, linear	English
MathQA	37200	+, -, /, $\times$	probability, physics, geometry, gain-loss	English
Math23K	23161	+, -, /, $\times$	algebra, linear	Chinese

3.2. Turkish MWP benchmark datasets

Creating an MWP dataset from scratch normally requires collecting a significant number of questions from websites or textbooks, forming the equations corresponding to the questions, and serving the data in a suitable format to train the machine learning models. Instead of this process, we preferred to translate existing English benchmark MWP datasets into Turkish. Translating English datasets into other languages using high-quality machine translation applications such as Google Translate [2] and Amazon Translate [3] is a recent research trend used in several NLP fields [27]. The machine translation approach makes it easy to create datasets and work on

[2]https://translate.google.com/
[3]https://aws.amazon.com/translate/

the MWP task, even for languages that do not have an MWP dataset yet. The process yields translations with acceptable quality and the translated dataset is either used as is or subjected to further manual postprocessing to increase its quality. Using an existing dataset translated from one language to another saves time and provides the opportunity for multilingual systems. In this respect, we translated MAWPS, ASDiv-A, SVAMP, and MathQA datasets into Turkish using Googletrans [4], a Python library that implements Google Translate API. We examined different machine translation systems and observed that Google produces the best results.

We generated two distinct datasets which differ in their complexity levels. The details of the compilation process are explained in Sections 3.2.1 and 3.2.2. Within the scope of the MWP task, the two datasets are the first comprehensive benchmark datasets created in Turkish and are aimed to contribute to future studies.

3.2.1. Combined dataset from MAWPS, ASDiv-A, and SVAMP

The first dataset is formed by combining the MAWPS, ASDiv-A, and SVAMP datasets, all containing elementary-level arithmetic questions. These three datasets are chosen to merge since the question sentences do not require extensive processing, the question types are comparable, and the mathematical equations are structured. In total, 4163 MWP data are provided.

Only the question texts in the datasets are translated. The numbers in the questions are replaced with *numberX* tags, where X is the order of the number among all the numbers in the question. Tagging the numbers with *numberX* tags provides a generic structure. When numbers are used as in the original datasets, each different number corresponds to a different embedding in the embedding space and it makes it difficult for the model to learn the common properties of the numbers. However, when the numbers are replaced with *numberX* tags that appear in all the questions, the attention mechanism learns to pay attention to these tokens as crucial points in the sentences. Therefore, the numbers in each question text are extracted and stored as a new field, then they are changed with tags according to their orders. When the preprocessing is completed for all the questions, the question texts are translated into Turkish. After translation, people names in the questions are manually replaced with Turkish names in order to better reflect the characteristics of a Turkish dataset. Lastly, a white space character is added before and after all punctuation marks so that each punctuation mark is considered a separate token.

A common template structure is used for the equations in the dataset. The equations in the original datasets are in *infix notation*, where the mathematical operators lie between the numbers (operands), as in "$(A + B) \times (C - D)$". We convert the equations into *prefix notation*, where the operator to be applied to two operands is given just before these operands, as in "$\times + A\ B - C\ D$". Finally, we build the dataset in JSON format with the fields "Question", "Equation", "Numbers", and "Answer". An example question is shown in Table 2.

Since the dataset we build contains questions originated from English elementary school-level textbooks, we also checked whether they have a similar nature as the Turkish questions in similar levels. Table 3 shows three types of questions taken from a website of elementary-level Turkish questions and an example question corresponding to each one drawn from the dataset. We see that the questions in the dataset are similar in terms of the difficulty level and the problem structure as the questions used in Turkish elementary school-level.

[4]https://pypi.org/project/googletrans [accessed on December 2021]
[5]https://www.ilkokuldokumanlari.com/3-sinif-toplama-islemi-problemleri/
[6]https://www.ilkokuldokumanlari.com/2-sinif-tartma-problemleri/
[7]https://www.ilkokuldokumanlari.com/3-sinif-bolme-problemleri/

Table 2. A sample from the combined dataset. The English translation of the question text is also shown in parenthesis.

Question	ezgi ' nin en sevdiği grup , biletlerin her birinin number0 dolar olduğu bir konser veriyordu . ezgi kendisi ve arkadaşları için number1 bilet ve başka biri gitmek isterse diye fazladan number2 bilet aldı . ne kadar harcadı ? (Ezgi's favorite band is giving a concert where tickets are number0 dollars each. Ezgi bought number1 tickets for herself and her friends and number2 extra tickets in case anyone else wants to go. How many dollars did she spend?)
Equation	× number0 + number1 number2
Numbers	4, 3, 5
Answer	32

Table 3. Comparison of questions in Turkish resources (left) and in the combined dataset (right). The English translations of the questions are also shown in parentheses. The English translations of the questions in the combined dataset are taken from the corresponding English dataset.

Original Turkish questions	**Translated Turkish questions**
Ali 18 yaşındadır. Ablası Ali'den 15 yaş büyüktür. Ali ile ablasının yaşları toplamı kaçtır? [5] (Ali is 18 years old. His sister is 15 years older than Ali. What is the sum of the ages of Ali and his sister?)	Ben 6, annem ise 26 yaşındadır. Babamın yaşı, annem ile benim yaşımın toplamı kadardır. Yaşlarımızın toplamı kaçtır? (I am 6 years old and my mother is 26 years old. My father's age is the sum of my mother's age and my age. What is the sum of our ages?)
Bir manav 80 kg elmanın sabah 27, öğleden sonra 17 kilogramını sattı. Manavda kaç kilogram elma kaldı? [6] (A fruit seller sold 27 kg of 80 kg of apples in the morning and 17 kg in the afternoon. How many kg of apples are left in the fruit seller?)	Davut 18 bilet kazanmıştı. 5 tanesini oyuncak almak için ve 11 tanesini de giysi almak için kullandıysa, kaç bileti kalmıştır? (Dave had won 18 tickets. If he used 5 to buy some toys and 11 more to buy some clothes how many tickets did Dave have left?)
Bir otomobil 5 litre benzinle 60 kilometre yol gitmektedir. Bu araç 8 litre benzinle kaç km yol gider? [7] (A car travels 60 kilometers on 5 liters of gasoline. How many kilometers does the car travel on 8 liters of gasoline?)	Arabam litre başına 20 kilometre yol alıyor. 5 litre benzinle kaç kilometre gidebilirim? (My car gets 20.0 miles per gallon. How many miles can I drive on 5.0 gallons of gas?)

3.2.2. MathQA dataset

The MathQA benchmark dataset consisting of 37,200 instances is used as the second dataset. This dataset is chosen because it is one of the most challenging datasets, the amount of data is satisfactory, and it covers a variety of questions from many fields. The dataset requires much more processing than the MAWPS, ASDiv-A, and SVAMP datasets, as the questions and answers are collected from websites. We manually processed the entire dataset as detailed below.

In the original dataset, an annotated equation template is used to formulate the equations, an example of which is shown in Table 4. However, there are unknowns, constant terms, and complex mathematical operations in the annotated equations. Also, the numerical answers in the dataset are not always the same as the result of the annotated formula. To obtain a standard and more convenient form for the equations, we converted the annotated formulas into prefix notation as in the previous dataset. The constants in the form of *const_X* (e.g., const_100 used in percentile problems) are replaced with their numerical forms. The mathematical operators in the spelled form are converted into the mathematical form. Table 4 shows the converted equation for the given annotated equation.

We removed some questions from the dataset that require advanced operations such as round and log apart from the four basic mathematical operations to be solved. We also deleted the instances with more than

Table 4. An example equation in the original dataset and the translated dataset.

Annotated equation template	multiply(divide(80, 160), const_100)
Converted equation template	* / 80 160 100

15 tokens in the equation part. Some of the problems in the domain of physics, geometry, probability, and economics that require knowledge of external formulas and that have equations with many unknowns were eliminated. Apart from these, the instances that have unknown characters in the question text were removed. The reason of these simplification operations is that we focus primarily on the solution of elementary school-level MWPs in this work and we leave the problems that require more sophisticated solutions as future work. After the manual analysis of the dataset, it was reduced to 19,555 data instances.

Moreover, several processing operations are carried out before and after the translation. Although there is a large number of operations performed, we include here only the important ones to serve as a guideline for future studies to translate datasets between different languages.

- The ordinal numbers are expressed in written form.
- English abbreviations in the question texts are written in their full forms since the Googletrans API cannot translate the abbreviations properly.
- The numerical terms such as “two”, “a hundred”, and “million” are converted into numbers.
- Sentences starting with “find”/“solve”/“calculate” keywords but ending with a question mark, and sentences starting with “what”/“how” keywords but ending with a period are edited.
- The commas used as the thousand separators in English are removed. The reason is that Googletrans API translates the comma, which is the thousands separator, and the period, which is the decimal separator, as a period.
- Multiplication, division, and exponentiation operators expressed with different symbols are converted to a standard form.
- The different Unicode characters used in place of apostrophes, double quotes, and dashes are adjusted.

As in the first dataset, we checked whether the questions in the MathQA dataset conform to the question style used in Turkish elementary-level questions. Since the questions in this dataset are gathered from online forums and websites, they are more diverse than those in the other dataset. We tried to limit the question types in the dataset to elementary school-level questions by simplification operations mentioned above. However, it is not possible to say that all the questions in the dataset are as easy and appropriate for that level as in the first dataset. In this respect, it can be regarded as a challenging dataset for the MWP solving task. Table 5 shows three Turkish questions as examples taken from maths-related websites and a corresponding question for each from the translated MathQA dataset.

[8] https://eodev.com/gorev/8894969

[9] https://www.matematikkolay.net/oran-oranti/aritmetik-ortalama

[10] https://matematikdelisi.com/Orta/Sinif5/Test/ondalik-gosterim/ondalik-gosterim-konu-testi-3.html

Table 5. Comparison of questions in Turkish resources (left) and in the MathQA dataset (right). The English translations of the questions are also shown in parentheses. The English translations of the questions in the MathQA dataset are taken from the corresponding English dataset.

Original Turkish questions	**Translated Turkish questions**
Bir sayının 18'e bölümünden kalan 7'dir. Aynı sayının 6 ile bölümünden kalan kaçtır? [8] (The remainder after dividing a number by 18 is 7. What is the remainder after dividing the same number by 6?)	Bir sayı 84 ile bölümünden 57 kalanını verir. Aynı sayının 12 ile bölümünden kalan kaçtır? (A number divided by 84 leaves remainder 57. What is the remainder when the same number is divided by 12?)
Yaş ortalaması 30 olan 12 kişiden yaşları 32 ve 28 olan iki kişi ayrılıyor. Yeni ortalama kaç olur? [9] (Two people aged 32 and 28 are separated from 12 people whose average age is 30. What is the new average?)	Bir gruptaki 36 öğrencinin yaş ortalaması 14'tür. Buna öğretmenin yaşı da dahil edildiğinde ortalama 1 artıyor. Öğretmenin yaşı kaçtır? (The average age of 36 students in a group is 14 years. When the teacher's age is included to it, the average increases by 1. What is the teacher's age in years?)
8,a37 sayısının onda birler ve yüzde birler basamaklarındaki rakamların toplamı 10 ise, a kaçtır? [10] (If the sum of the digits in the tenths place and the hundredths place of the number 8.a37 is 10, what is a?)	6/11 ondalık biçiminde ondalık noktanın sağındaki 26. basamak nedir? (What is the 26th digit to the right of the decimal point in the decimal form of 6/11?)

4. Methodology

The MWP solving system proposed in this work consists of the steps of processing the input, feeding the input to the embedding model, encoding the embedded question text in the encoder, and generating the equation in the decoder. The details of the process are explained in the following sections.

4.1. Data processing

We first tokenize the question texts and the equations in the datasets. Then, we create a dictionary of questions from different tokens in the question texts and a dictionary of equations from different tokens in the equations, separately for both datasets. For each token, the question vocabulary includes a unique identifier, the token itself, and the number of times the token occurs in the questions. The special tokens <s>, </s>, and <unk> are used to denote, respectively, the beginning of a question/equation, the end of a question/equation, and a placeholder for the unknown words if needed. The equation vocabulary is formed in the same manner, including the operators, constant numbers, and *numberX* tags in the equations. The sizes of the question vocabulary and the equation vocabulary are 7252 and 29, respectively. Table 6 shows a sample from each vocabulary.

Table 6. Question and equation vocabularies.

Id	**Token**	**Count**		**Id**	**Token**	**Count**
0	<unk>	1		0	<unk>	1
1	<s>	1		1	<s>	1
2	</s>	1		2	</s>	1
3	ali	19		3	+	1916
4	number0	4245		4	number0	3991
5	şekeri	37		5	number1	3820
...	...	...		...	...	...
7250	tarifleri	1		27	5	2
7251	gereklidir	1		28	10	1

4.2. Embedding models

There are different embedding models used in NLP applications to represent words with low-dimensional vectors. These models capture the context of the words, the syntactic and semantic similarities of the words, and the relationship of the words to other words [28].

In this work, we make a comprehensive analysis by using many different word embedding models and pretrained language models and observe their performance on the Turkish MWP task. As static embeddings, we use Word2vec, GloVe, and fastText embeddings. The embeddings for unknown words in word2vec and GloVe models are initialized randomly using the standard normal distribution with zero mean and unit variance. Static embedding models have the drawback that they cannot distinguish different senses of a word and use the same embedding vector independent of the sense in which the word is used. To alleviate this problem, contextual embedding methods are used which mostly show better performance than the static methods. As contextual embeddings, we use BERT, ELECTRA, and ConvBERT embeddings. In the BERT model, we add the [CLS] and [SEP] tokens to the text and pad the text with the special [PAD] token so that all text sequences have the same length as the longest sequence in the batch. We utilize an attention mask to prevent the model from weighting the [PAD] token in the sequence. We give the text to the BERT encoder and obtain the contextual embeddings of the tokens in the text. The processing stages for the ELECTRA and ConvBERT models are similar as for BERT.

4.3. Encoder and decoder models

The architecture of the seq2seq model used in this work is shown in Figure 2. The encoder is a 2-layer bidirectional RNN model. We use a bidirectional model in order to be able to address both future and past contexts during encoding. We employ both LSTM and GRU networks in the encoder and compare their performance. The tokens in the question texts are first mapped to word embeddings using one of the embedding models depicted in Section 4.2. The embedded question is then fed to the encoder. The encoder yields both outputs and hidden vectors that will be used during decoding.

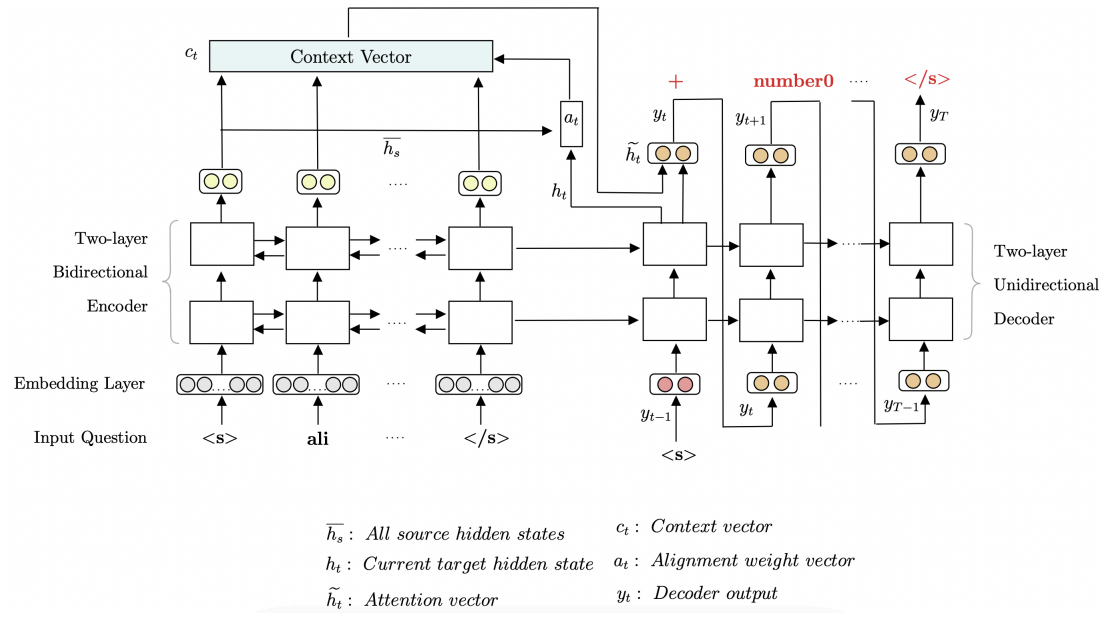


Figure 2. Overall architecture of the MWP model.

The decoder is a unidirectional RNN with an attention mechanism. In seq2seq models, the attention mechanism is frequently used to learn the alignments between the input and output sequences. We use the Luong attention mechanism [29], which basically differs from the classical Bahdanau attention [30] by using the current decoder hidden state rather than the previous hidden state in attending to the input sequence. As in the encoder, we use both LSTM and GRU models as the recurrent network in the decoder.

The decoding process begins by feeding the <s> tag as the first input token to the decoder. At time step t during decoding, the current decoder hidden state, the previous output embedding (the correct output vector during training and the estimated output vector during testing), and the context vector are used to obtain the current output. The context vector weighs the contribution of each input word by using the encoder hidden states in generating the current output. The equation is generated one token at a time until the </s> token is produced as the decoder output.

5. Experiments and results

In this section, we first give some statistics about the datasets and indicate the success of the translation process. We then show the results of the experiments with different encoder-decoder models and word embedding approaches. This is followed by a comparison of the results with those of the neural MWP studies in the literature. Finally, we explain the hyperparameters used and the implementation details.

5.1. Dataset statistics

We split each dataset as 80% for training and 20% for testing randomly as shown in Table 7. Figure 3 is a histogram plot of the number of tokens in the question texts in the datasets. We note that, although the datasets are randomly divided into training and test splits, the MathQA dataset has a different distribution in the two splits with respect to the question length. This makes the dataset more challenging and may affect the success rates of the experiments adversely, which is indeed the case as will be observed in Section 5.3. Similarly, Figure 4 shows the number of tokens in the equations in the datasets.

Table 7. Train and test split sizes of the datasets.

Dataset	# Instances in train set	# Instances in test set
Combined	3301	862
MathQA	15,651	3904

5.2. Machine translation evaluation

Since the datasets built in this research are formed by automatic translation, it is important to evaluate their quality before using them for training the MWP systems. For this purpose, we randomly selected 20 question texts from the English MAWPS, ASDiv-A, and SVAMP datasets and 20 question texts from the English MathQA dataset. The question texts were manually translated into Turkish by one of the authors of this paper. The automatic translations were then evaluated using manual translations as the reference translations. The BLEU scores are 91.46% and 94.89% for the combined dataset and the MathQA dataset, respectively. It indicates that the automatic translations are very close to the human translations.

Although creating datasets in a low-resource language with machine translation provides great convenience, there are also some limitations. For instance, foreign person names remain as they are in the original

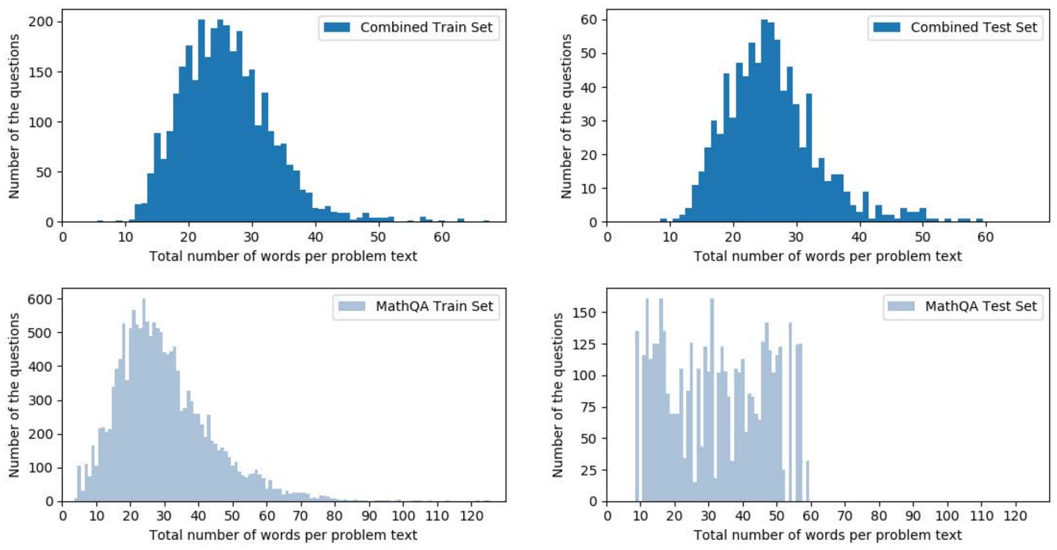


Figure 3. Number of tokens in the questions in the two datasets.

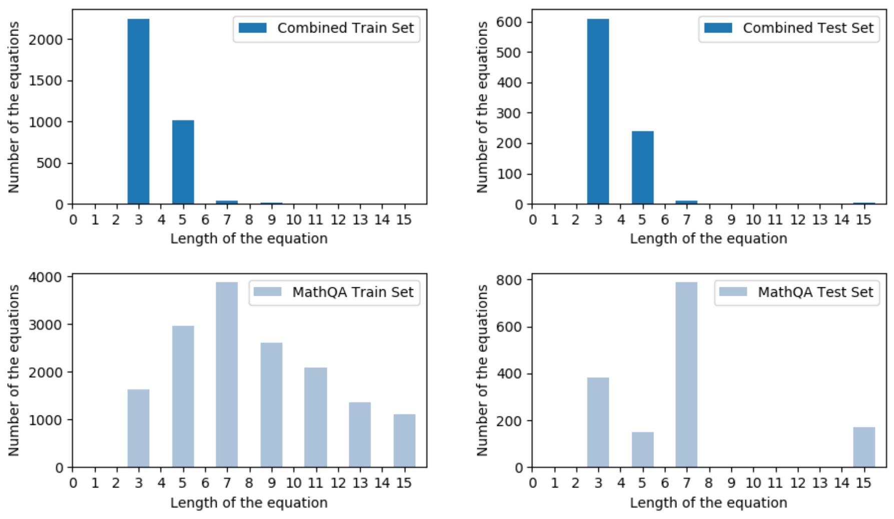


Figure 4. Number of tokens in the equations in the two datasets.

dataset. However, it is better to use language-specific names in the target language dataset for consistency. In addition, local terms, idioms, multiword expressions, technical terms, and abbreviations used in the original language cannot always be properly translated to the target language. Similarly, local units of measurement are not converted to their equivalents. To handle such problems, some preprocessing and postprocessing operations should be applied to the source and target datasets. This is also the approach we followed as explained in Section 3.2.

5.3. Results of the models

The models are evaluated with two different evaluation metrics, which are the BLEU-4 score and the accuracy metric. The accuracy is computed as the ratio of the number of test instances where the equation produced by the model and the corresponding reference equation are precisely the same to the total number of test instances.

In some cases, the model output and the reference equation may not be exactly the same, but the tokens may highly overlap. In order to take such cases into account, we use the BLEU-4 score as a second evaluation metric which computes a score based on overlapping n-grams (n=1,...,4) between the two equations.

As stated in Section 4.2, we represent the input tokens in the question texts and the equations using different types of embeddings. We use the static embedding models word2vec, GloVe, and fastText and the base uncased Turkish contextual embedding models BERTurk [11], ConvBERTurk [12], and ELECTRA [13]. We train the seq2seq model with each embedding approach separately and observe the performance on the test set. We use the embedding vector dimensions recommended by the word embedding libraries used in this work as shown in Table 8.

Table 9 shows the results of the experiments for the combined dataset. We first employed the GRU encoder-decoder model with different number of layers and hidden unit sizes in the encoder and decoder. We see that the performance of the model increases as the layer number and the hidden vector size increase with all embedding types. There is especially a large margin between the best model (2 layers and 256 hidden units) and the other GRU models. These results indicate that a 1-layer encoder-decoder network with a small hidden vector size cannot sufficiently capture the information within the questions and produce the correct equation tokens. We also tested the performance with the LSTM encoder-decoder model as the second type of RNN architecture. As in the case of the GRU model, the best results were obtained with 2 layers and 256 hidden units. We also experimented with different parameters as the layer number and hidden vector size, but they showed worse performance.

Table 8. Embedding vector dimensions for the static and contextual embedding methods.

Method	**Embedding dimension**
Word2vec	400
GloVe	300
fastText	300
BERT	768
ConvBERT	768
ELECTRA	256

Table 9. Performance results of the models on the combined dataset.

	BERT		**ConvBERT**		**Word2vec**		**GloVe**		**fastText**		**ELECTRA**	
	Acc.	**BLEU**	**Acc.**	**BLEU**	**Acc.**	**BLEU**	**Acc.**	**BLEU**	**Acc.**	**BLEU**	**Acc.**	**BLEU**
GRU model												
1 layer, 128 hidden units	65.78	67.97	67.05	69.45	57.77	60.71	57.42	59.76	40.72	49.84	57.54	60.13
1 layer, 256 hidden units	67.98	69.12	69.14	69.04	59.63	60.87	58.82	61.64	46.40	51.51	59.86	63.04
2 layers, 128 hidden units	69.25	69.31	69.26	70.61	60.21	63.74	60.56	63.89	47.91	53.73	59.28	63.07
2 layers, 256 hidden units	**72.34**	**73.70**	69.49	71.22	63.11	66.24	61.72	64.47	51.62	57.62	60.21	64.28
LSTM model												
2 layers, 256 hidden units	**70.30**	**70.94**	67.05	68.40	62.41	63.87	61.25	63.33	58.82	62.862	63.11	64.76

As the best results on the combined dataset, we obtained around 72% accuracy and 73% BLEU score. We observe that contextual embeddings give rise to much better success rates than static embeddings. This is

[11] https://huggingface.co/dbmdz/bert-base-turkish-128k-uncased
[12] https://huggingface.co/dbmdz/convbert-base-turkish-mc4-uncased
[13] https://huggingface.co/dbmdz/electra-base-turkish-mc4-uncased-generator

especially the case for the BERT and ConvBERT embeddings and the BERT embeddings yield the best overall performance with both GRU and LSTM models. We also performed an error analysis on the outputs of the models. As examples of error cases, Table 10 shows some instances where the models give incorrect outputs.

We also conducted a cross-lingual experiment in order to see whether the use of in-language resources is necessary to obtain high performance or simply using English resources yields sufficient performance. In this respect, we trained the model on the training set of the combined English MAWPS, ASDiv-A, and SVAMP datasets using the multilingual BERT (mBERT) [14] embeddings. For testing, we used the test set of the Turkish combined dataset. We obtained an accuracy of 32.71% and a BLEU score of 46.17% by using the 2-layer GRU and 256 hidden units parameters, which are quite low compared to the scores in Table 9. The low performance of the cross-lingual experiment indicates that the use of in-language resources contributes significantly to the success of the models.

Table 10. Example instances from combined dataset with incorrect predictions. The English translations of the questions are taken from the corresponding English dataset.

Question	Esma, araba yıkamaktan number0 çeyrek tasarruf etti. Esma'nın kaç senti var? (Sara has saved number0 quarters from washing cars. How many cents does Sara have?)
Equation	* number0 25
Predicted equation	* number0 12
Reason of error	Failure to make inferences about currency conversions
Question	Erol evin duvarları için number0 büyük tahta kalas kullandı. Her bir kalasın sabitlenmesi için number1 çiviye ihtiyacı varsa ve ek olarak bazı küçük kalaslar için number2 çiviye ihtiyacı varsa, Erol'un evin duvarı için kaç çiviye ihtiyacı var? (For the walls of the house, John would use number0 large planks of wood. If each plank of wood needs number1 pieces of nails to be secured and in addition he needs number2 nails for some smaller planks, how many nails does John need for the house wall?)
Equation	+ * number0 number1 number2
Predicted equation	* number0 + number1 number2
Reason of error	Failure in groups of items

Despite the success of the models on the combined dataset, we could not obtain good performance results on the MathQA dataset. The 2-layer GRU model with 256 hidden units showed an accuracy of 18.69% and a BLEU score of 32.32%, while the 2-layer LSTM model with 256 hidden units showed 18.29% accuracy and 32.20% BLEU score. The low performance on this dataset can be attributed to the wide variety of question types, different patterns in the questions, and inconsistencies between the questions and equations.

5.4. Comparison with Other MWP studies

Since there are no Turkish MWP studies that make use of neural models in the literature, we compare our results with those of the studies on the English datasets. We should note that there are significant differences in terms of experimental setup between our study and the other studies. First, the languages are different and we work on a language that has different morphological and syntactic characteristics than English. Second, although the Turkish datasets are obtained as translations of the English datasets, the sizes and slightly the contents of the datasets are different due to the preprocessing and postprocessing operations explained in Section 3.2. Third,

[14] https://huggingface.co/bert-base-multilingual-cased

we do not apply cross-validation as in some of the other studies since we aim at having a fixed split that will allow comparisons between future Turkish MWP studies. Hence, the performance results given in this section should be read and compared with these differences in mind.

Amini et al. [26] measure the performance of the English MathQA dataset by using a sequence-to-program model, which provides an additional layer of supervision to limit the impact of statistical bias in the dataset. They extend their model to integrate domain knowledge for different problem categories like geometry and probability by modifying the RNN decoder layers. A different set of parameters is used for each category during hidden state computations and this is achieved by a hard switch mechanism in the neural network. They obtained a test accuracy of 51.9%. The success rates obtained in this study shows that using more dynamic models in solving MWPs containing questions from many domains increases performance. Patel et al. [24] propose a seq2seq model with RoBERTa pretrained embeddings as input. Using the MAWPS and ASDiv-A datasets for training and the SVAMP dataset for testing, they achieved 40.3% accuracy. Since SVAMP is a more challenging dataset than the other two, utilizing it directly in the test decreased the accuracy. In our work, we followed a different strategy by combining and then shuffling the three datasets to make the train and test sets more homogeneous. Lan et al. [31] implement a seq2seq LSTM encoder-decoder model and make separate analyses on the MAWPS, ASDiv-A, and SVAMP datasets. Table 11 summarizes the comparisons with the previous studies.

Table 11. Comparison with English MWP studies.

Study	**Dataset**	**Test accuracy**
Amini et al. [26]	English MathQA	51.9%
Patel et al. [24]	English MAWPS and ASDiv-A (train), SVAMP (test)	40.3%
Lan et al. [31]	English MAWPS	79.7%
Lan et al. [31]	English ASDiv-A	55.5%
Lan et al. [31]	English SVAMP	24.2%
Our study	Turkish MathQA	18.7%
Our study	Combined Turkish MAWPS, ASDiv-A, and SVAMP	72.34%

5.5. Hyperparameters and implementation details

The best results in this work were obtained using the BERT embedding model with 768 units, a 2-layer bidirectional GRU with 256 hidden units as the encoder, and a 2-layer unidirectional GRU with 256 hidden units as the decoder. Adam optimizer was used with a learning rate of 2e-4 for the question embeddings and a learning rate of 8e-6 for the equation embeddings. We applied standard dropout during training after each RNN layer with a probability of 0.1. We trained the model with a batch size of 8 over 50 epochs. To avoid exploding gradients, we used the gradient clipping technique by the norm. We calculated a negative log-likelihood loss over the decoder outputs. We added the softmax activation function to the output layer of the decoder. The experiments were conducted on a high-performance computing cluster at Boğaziçi University TETAM [15]. The models were developed using the PyTorch library in Python.

6. Conclusion

Solving math word problems is a widely studied NLP task in the English language. In Turkish, there are only a few studies that are based on rule-based models and there are no studies that use neural models. To remedy

[15]https://tetam.boun.edu.tr/ [accessed on September 2021]

this shortcoming, in this paper, we introduced a sequence-to-sequence neural model with the Luong attention mechanism for solving Turkish MWPs. We employed two different types of recurrent models, which are GRU and LSTM, in the encoder and the decoder, and compared these models with different layer numbers and hidden vector sizes. In addition to model comparisons, we made a detailed analysis of using different types of static and contextual embeddings as input to the models.

We also contributed to Turkish MWP studies by publishing two new Turkish MWP benchmark datasets. The datasets were obtained by automatic translation from English datasets. Several processing operations were applied before and after the translation to increase the quality of the datasets. The first dataset was compiled from the MAWPS, ASDiv-A, and SVAMP datasets and the second dataset was compiled from the MathQA dataset. We evaluated the quality of the translations using a sample of question texts, which showed that the automatic translations are very close to the manual translations.

In the experiments, we achieved 72.34% accuracy and 73.70% BLEU score on the combined dataset, which can be regarded as a successful result considering similar studies on the corresponding English datasets. However, the results on the second dataset are not as high as expected, which is a more challenging dataset. Despite the low accuracies on this dataset, it can be used as a benchmark dataset in future Turkish MWP studies.

As future work, we will explore several techniques to improve the results on the MathQA dataset, implement different neural models such as transformers, and integrate Bahdanau's attention mechanism [30]. We also plan to integrate into the models the commonsense knowledge we use while solving problems through semantic networks.

Acknowledgments

This work was supported by Boğaziçi University Research Fund under the grant number 16903. We would like to thank Boğaziçi University TETAM for providing access to the computing clusters to train and test our models.

References

[1] Bobrow D. Natural language input for a computer problem solving system. Massachusetts Institute of Technology, 1964.

[2] Mukherjee A, Garain U. A review of methods for automatic understanding of natural language mathematical problems. Artificial Intelligence Review 2008; 29 (2): 93–122.

[3] Bakman Y. Robust understanding of word problems with extraneous information. arXiv preprint, 2007. arXiv: 0701393v1.

[4] Say C, Akın L, Özsoy S. A program which solves arithmetic problems in Turkish. In: Proceedings of the Ninth International Symposium on Computer and Information Sciences; 1994. pp. 550–557.

[5] Say C. Understanding arithmetic problems in Turkish. International Journal of Pattern Recognition and Artificial Intelligence 2001; 15 (2): 359–374.

[6] Çakıroğlu Ü. Can computer understand and solve Turkish arithmetic problems?. World Applied Sciences Journal 2008; 5 (3): 311-317.

[7] Eken S, Ekinci E, Sayar A. Understanding and solving Turkish arithmetic problems via xml keywords. Düzce Üniversitesi Bilim ve Teknoloji Dergisi 2014; 2 (1): 48–55.

[8] Sutskever I, Vinyals O, Le Q. Sequence to sequence learning with neural networks. In: Proceedings of the International Conference on Neural Information Processing Systems; Cambridge, MA, USA; 2014. pp. 3104–3112.

[9] Liguda C, Pfeiffer T. Modeling math word problems with augmented semantic networks. In: Proceedings of the International Conference on Applications of Natural Language Processing to Information Systems; Berlin, Heidelberg, Germany; 2012. pp. 247–252.

[10] Yuhui M, Ying Z, Guangzuo C, Yun R, Ronghuai H. Frame-based calculus of solving arithmetic multi-step addition and subtraction word problems. In: Second International Workshop on Education Technology and Computer Science; 2010. pp. 476–479. doi: 10.1109/ETCS.2010.316

[11] Hosseini M, Hajishirzi H, Etzioni O, Kushman N. Learning to solve arithmetic word problems with verb categorization. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing; Doha, Qatar; 2014. pp. 523–533.

[12] Kushman N, Zettlemoyer L, Barzilay R, Artzi Y. Learning to automatically solve algebra word problems. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics; Baltimore, Maryland, USA; 2014. pp. 271–281. doi: 10.3115/v1/P14-1026

[13] Mitra A, Baral C. Learning to use formulas to solve simple arithmetic problems. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics; Berlin, Germany; 2016. pp. 2144–2153.

[14] Upadhyay S, Chang M, Chang K, Yih W. Learning from explicit and implicit supervision jointly for algebra word problems. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing; Austin, Texas, USA; 2016. pp. 297–306.

[15] Wang Y, Liu X, Shi S. Deep neural solver for math word problems. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing; Copenhagen, Denmark; 2017. pp. 845–854.

[16] Robaidek B, Koncel-Kedziorski R, Hajishirzi H. Data-driven methods for solving algebra word problems. arXiv preprint, 2018. arXiv: 1804.10718.

[17] Huang D, Yao J, Lin C, Zhou Q, Yin J. Using intermediate representations to solve math word problems. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics; Melbourne, Australia; 2018. pp. 419–428.

[18] Huang D, Liu J, Lin C, Yin J. Neural math word problem solver with reinforcement learning. In: Proceedings of the 27th International Conference on Computational Linguistics; Santa Fe, New Mexico, USA; 2018. pp. 213–223.

[19] Sachan M, Xing E. Learning to solve geometry problems from natural language demonstrations in textbooks. In: Proceedings of the 6th Joint Conference on Lexical and Computational Semantics; Vancouver, Canada; 2017. pp. 251–261.

[20] Seo M, Hajishirzi H, Farhadi A, Etzioni O, Malcolm C. Solving geometry problems: combining text and diagram interpretation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing; Lisbon, Portugal; 2015. pp. 1466–1476.

[21] Huang D, Shi S, Lin C, Yin J, Ma W. How well do computers solve math word problems? large-scale dataset construction and evaluation. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics; Berlin, Germany; 2016. pp. 887–896.

[22] Wang H, Tian F, Gao B, Zhu C, Bian J et al. Solving verbal questions in IQ test by knowledge-powered word embedding. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing; Austin, Texas, USA; 2016. pp. 541–550.

[23] Koncel-Kedziorski R, Roy S, Amini A, Kushman N, Hajishirzi H. MAWPS: a math word problem repository. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; San Diego, California, USA; 2016. pp. 1152–1157.

[24] Patel A, Bhattamishra S, Goyal N. Are NLP models really able to solve simple math word problems? In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2021. pp. 2080–2094.

[25] Miao S, Liang C, Su K. A diverse corpus for evaluating and developing English math word problem solvers. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics; 2020. pp. 975–984.

[26] Amini A, Gabriel S, Lin P, Koncel-Kedziorski R, Choi Y et al. MathQA: towards interpretable math word problem solving with operation-based formalisms. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; Minneapolis, Minnesota, USA; 2019. pp. 2357–2367.

[27] Budur E, Özçelik R, Güngör T, Potts C. Data and Representation for Turkish Natural Language Inference. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing; 2020:8253–8267.

[28] Naseem U, Razzak I, Khan S, Prasad M. A comprehensive survey on word representation models: from classical to state-of-the-art word representation language models. ACM Transactions on Asian and Low-Resource Language Information Processing 2021; 20 (5): 1–35.

[29] Luong M, Pham H, Manning C. Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing; 2015. pp. 1412–1421.

[30] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. In: 3rd International Conference on Learning Representations; San Diego, CA, USA; 2015.

[31] Lan Y, Wang L, Zhang Q, Lan Y, Dai B et al. Mwptoolkit: an open-source framework for deep learning-based math word problem solvers. arXiv preprint, 2021. arXiv: 2109.00799.