Simple Mathematical Word Problems Solving with Deep Learning

Sizhu Cheng
ICME
Stanford University
Palo Alto, CA 94305
scheng72@stanford.edu

Nicolas Chung
SCPD
Stanford University
Palo Alto, CA 94305
sunchung@stanford.edu

Abstract

People use their knowledge to solve extensive mathematical problem every day in real life. Mathematical problems are often stated in words in different scenarios, thus requiring problem solvers to extract information from the text and formulate in mathematical language to get the problem's answer. We delved into simple algebra math problems and experimented with different deep learning model including bidirectional GRU seq2seq models and its variants, as well as Transformer. Experiments show that transformer outputs the baseline RNN model in this task, for its ability to set up valid equations coherent to the mathematical logics.

1 Introduction

Intelligence cannot be deprived of mathematical reasoning. Mathematical problems solving is integrated into people's life when they calculate ingredients for recipes, changes in purchase and taxes etc. Because of this, machine solving mathematical word problems (MWPs) has been an interesting problem for scientists for more than half of the decades [1]. Before 2010, most of the efforts have been put into primarily in the rule-based methods only [6, 11]. With the boom of the machine learning starting from 2010, scientists start to explore the statistic methods,including the semantic parsing and feature engineering, in the field of MWPs study. In the recent year, due to the availability of large-scale training sets and increasing computation power, deep learning becomes another direction to attempt [11].

MWPs solving is believed to be challenging because of the semantic gap between the mathematical expressions and language logic [5]. The ordering in the mathematical expressions don't always matter as what is expected in simple text NLP problem because of commutative laws of addition and multiplication. Because of this, the state-of-the-art of MWPs only achieve 36% accuracy in a 5-choose-1 multiple choice task [5]. In this paper, we focused our work on arithmetic mathematical problems, and constructed models to output the mathematical equations given a chunk of MWPs in English text. We explored different deep learning methods with least amount of feature engineering.

Our works involves the four main steps. First, we preprocess our datasets. Second, three different models are built, including a bidirectional LSTM (encoder)-LSTM (decoder) attention (i.e. BiLLAtt) model, a bidirectional GRU (encoder)-LSTM (decoder) attention (i.e. BiGLAtt) model, and a transformer model. Third, we tune some hyperparameters. Last, we carried out qualitative analysis of our generated outputs to study the behavior of our model.

2 Related Work

Most of the work in MWPs using statistical methods with some feature engineering [4, 5, 10]. Currently, most of the published results are usually evaluated on combinations of several submodels,

instead of a single one and the state-of-art is still not satisfying. Kushman et al. [4] constructed a system of equation templates. At test time, information from MWPs are extracted and aligned with templates to instantiate the equations. Ling et al. [5] from DeepMind works on extracting rationals on more than 100,000 question-answer pairs based on an idea similar to semantic parsing. They managed to built a most likely latent system to capture the mathematical logic. However, the accuracy only achieves 36.4%, even with copy mechanism added.

Our two Recurrent Neural Network (RNN) models were inspired by the Deep Neural Solver (DNS) Model proposed by Wang et al. [10]. Wang et al. [10]'s task involves generating both equations and solutions from Chinese and English MWPs datasets using both deep learning and feature engineering. Deep Neural Solver consists of an inattentional bidirectional RNN seq2seq model, which uses GRU as the encoder and LSTM as the decoder to construct equation template, and a many-to-one simple RNN which identifies significant number and maps to equations [10]. Inspired by this, we borrowed their seq2seq part, but with multiplicative attention added, as our baseline model to stage our work.

Transformer Network is a model solely based on attention mechanism proposed by Vaswani et al. [8]. Without recurrence, it enables the computation more parallelizable. Transformer is believed to be one of the leading approaches in many NLP tasks including question answering, summarization and machine translation. However, not enough work has been done to MWPs using transformers, besides several recent papers exploring transformers on mathematical reasoning [7].

3 Approach

3.1 Baseline: Bidirectional GRU-LSTM (BiGLAtt) and LSTM-LSTM model (BiLLAtt)

We adopted this bidirectional unit from Wang et al. [10]'s work on DNS with additional attention as our baseline for this task. The main idea of the attention is to use a direct connection to the encoder at each step of the decoder to focus on a particular part of the source sequence. Adding attention enables source representations to be sensitive to the partial translation generated by the decoder of simple GRU-LSTM model to solve the "information bottleneck" at the last layer of the encoder. Besides, we also conduct experiments to replace the encoder GRU with LSTM in the above model and test its performance.

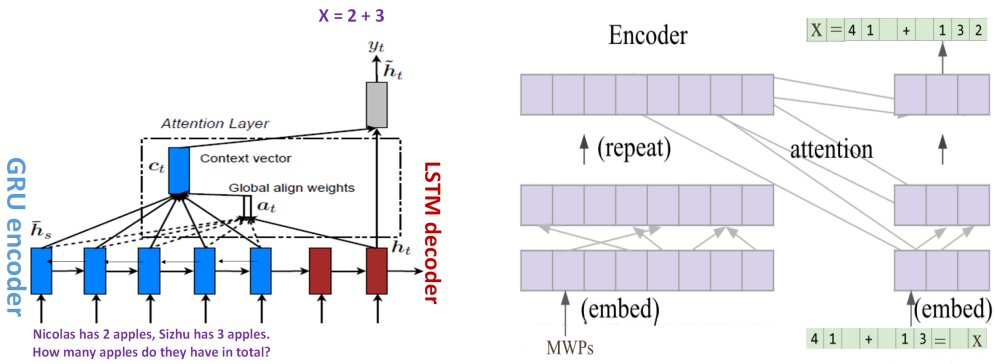
Our baseline model BiGLAtt, illustrated in **Figure 1a**, comprises of 3 main parts: a bidirectional GRU encoder, a unidirectional decoder and a multiplicative attention. The encoder is a RNN that takes in one word at a time of the input sequence, which is a math word problem in our case. The hidden states of encoder, h_t, will be used to compute multiplicative attention that produces an output a_t. Specifically, a_t is defined as follows:

$$a_t = \sum_{i}^{m} \alpha_{t,i} \mathbf{h_i}^{\text{dec}}$$

where $\alpha_t = \text{Softmax}((\mathbf{h_t^{dec}})^{\mathbf{T}} \mathbf{W}_{\text{attProj}} \mathbf{h_i^{enc}})$ for $i \in [m]$ as the index given to the tokens of source sentence. a_t will be concatenated with the hidden state of the decoder before generate the softmax distribution of the predicted word. In our case, we will have a math equation as our final output. Examples are provided in **Figure 1a** as well. BiLLAtt has a similar architecture as BiGLAtt, just replacing the GRU encoder with LSTM encoder instead.

3.2 Transformer

The idea of using transformer models come from identifying MWPs Solving as a seq2seq text-input problem. Some recent applications in mathematical reasoning also indicate the validity of using transformer to handle mathematics [7]. We recognize that building a problem solver for a word math problem shares similarities with machine translation in multiple ways. Both tasks involve seq2seq models, and formulating the math problems in equations is analogous to translation from English language into mathematical language. In light of these, we plan to incorporate transformer layers in our experiments and test them to see how they perform in MWPs Solving. Architectures of the transformer used here is shown in **Figure 1b**.



(a) Baseline model - BiGLAtt

(b) Transformer model

Figure 1: Graphical representations of baseline BiGLAtt and Transformer model. The BiLLAtt graph is omitted here. Note that the only difference between BiLLAtt model and BiGLAtt model is to replace the GRU encoder with LSTM decoder.

4 Experiments

4.1 Dataset and preprocessing

We collected data from different sources. The details of the dataset are described in **Table 1**.

Table 1: Our MWPs data from different sources

dataset name	number of math problems	source
Dolphin 18K	18460	Yahoo
MaWPS	3914	Math Word Problem Repository
AQUA-RAT	100000	Ling et al. [5]

[1] Dolphin 1878: from community question answering site - Yahoo
[2] MaWPS: from University of Washington
[3] AQUA-RAT: Multiple Choices Question from Ling et al. [5]

Math Word Problem Repository is a repository with extendable mathematical word problems [3]. It allows people to keep adding new single problem and equips with backend tool to select datasets with reduced lexical overlap. Each piece of data from MaWPS is a dictionary. It may have keys including index, alignments, equations, solutions and questions. However, most of data don't have all these attributes. We preprocessd the data obtained from MaWPS and collected all data with both questions and solutions to be used for our models.

Dolphin 18K is originated from community question answering site (Yahoo), containing 18,460 math problems. First, an open source code is used to scrape down all the MWPs listed in the dataset from `answers.yahoo.com` using Python library `urlib`. We then manually clean this dataset to obtain a reasonable amount of data.

Algebra Question Answering with Rationales Dataset (AQUA-RAT) is a dataset with about 100,000 math problems in multiple choices format collected by DeepMind Ling et al. [5]. To obtain the target corresponding equations, we set up a complex regular expressions in the format that

```
r'((?<!\n)[\[\(]?([a-z]|((\d*\.)?\d+))([ \)]?[-+/*=\^x]+
[ \(]?([a-z]|((\d*\.)?\d+))[\]\)]?)+(?=((\d*\.)?\d+))
([a-z]|((\d*\.)?\d+)*)+)'
```

forcing that

- variables and numerals can both appear in the equation. But in equations must have at least one float

- an equation sign must exist (Here we ignore all the examples with other Unicode Math symbols (e.g. $<$, $>$) only)

By this, we extracted almost all the equations from 'rationale' part in AQUA-RAT data. Note that these equations may not be the correct target equations we want. Since 'rationale' usually describes how people should brainstorm to obtain the final answer, it doesn't necessarily contain the corresponding equations.

We mix all training data from the three datasets and randomly shuffle them. We created a test set by combining the test set from AQUA-RAT and Dolphin 18k and randomly shuffling them as well (note that MaWPS is a single repository without train-val-test splitting). By this, we obtained 45446 piece of training data and 325 test data in total. Examples of how original data looks like can be found in **Appendix**.

4.2 Baseline: Simple Seq2Seq Recurrent Model

4.2.1 Experiment Details

In this approach, we split the data from MaWPS only into the following ratio: 3000 for training, 100 for dev and 100 for tests. We borrow the architecture assignment 4 and built from [2]. For the baseline models, we use the same number of hidden units and batch sizes from assignment 4. Some tuning has been done and the Corpus BLEU scores are shown in **Tables 2 and 3** and best BLEU score outputs plots are shown in **Figure 2 and 3**.

4.2.2 Results

Tuning / BLEU scores for BiGLAtt			
embed-size	hidden-size	dropout	Corpus BLEU
32	128	0.5	10.788860796543695
32	256	0.5	17.67595726
32	512	0.5	28.14589899
32	512	0.3	19.10077034
32	1024	0.3	24.05465502

Table 2: BiGLAtt Hyperparameter tunning and corresponding Blue Scores

Tuning / BLEU scores for BiLLAtt			
embed-size	hidden-size	dropout	Corpus BLEU
32	128	0.5	6.726031393
32	256	0.5	26.95214788
32	512	0.5	28.95965651
32	512	0.3	17.3544756
32	1024	0.3	22.80167961

Table 3: BiLLAtt Hyperparameter tunning and corresponding Blue Scores

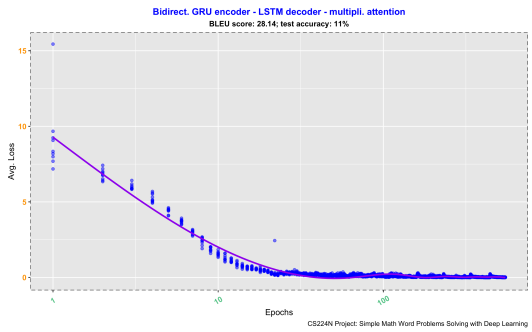


Figure 2: Baseline model BiGLAtt Training and validation loss in log 10 scale

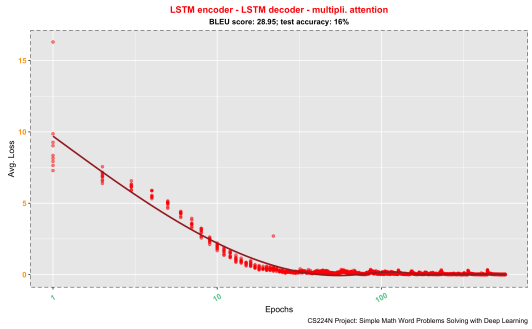


Figure 3: Baseline model BiLLAtt Training and validation loss in log 10 scale

In this baseline model, we observed that many <unk> tokens are generated. BiLLAtt outputs slightly less <unk> tokens than the BiGLAtt, but the number of <unk> tokens on test data is still significant.

4.3 Transformer

4.3.1 Experiment Design and Evaluation Method

In this approach, we split the preprocessed 45446 training data from all three datasets into 9:1 ratio each time and use 10-fold cross validation in our training. We use tensor2tensor (T2T) library, released and maintained by Google AI team, to perform all the experiments related with transformer [9]. The loss used in this model is the cross entropy loss, defined as

$$L = -\sum_{i=1} \mathbf{y_i} \log(\mathbf{p_i})$$

where $\mathbf{y_i}$ is the embedding vector for the true equation, $\mathbf{p_i}$ is the predicted distribution of the equations for i-th example.

We conducted our experiment using the similar settings recommended for large translation task. Here we train a light version of the transformer by setting hyperparameters equal to `transformer_base_single_gpu`. Specifically, the setting has hidden size 512, filter size 2048, hidden layers 6, dropout probability 0.2, and maximum length 256. We tuned the batch size from 4092 to 2048, with a learning rate constant increasing from 1 to 2. The transformer model was trained for 250,000 steps. The training will early stop if the loss for the eval (dev) set doesn't change for 10 steps. Note that a step here refers a gradient descent step. In other words,

$$\text{step} = \text{epoch}\frac{\text{batch size}}{\text{dataset size}}$$

Accuracy and negative log perplexity are chosen as the evaluation metrics at validation and test time.

To test the result, we will take all problems from the test data and use the `t2t-decode` module to decode our the MWPs using our trained transformer model.

4.3.2 Results

For translation task, the training run total 250,000 steps without reaching early stopping criteria. The training loss achieved 0.00084497215 after 250,000 steps. The cross entropy loss across all training step are shown in **Figure 4**. The graph of Negative Log Perplexity and Accuracy of validation set during training is shown in **Figure 5**. The two figures suggest that beyond 15.000 steps, the model started to overfitting and we should use the model obtained at 15,000 steps as our final model at test time. The training loss achieved 0.6697 at 15,000 steps. We summarize the quantitative metrics used for validation set as in **Table 4**.

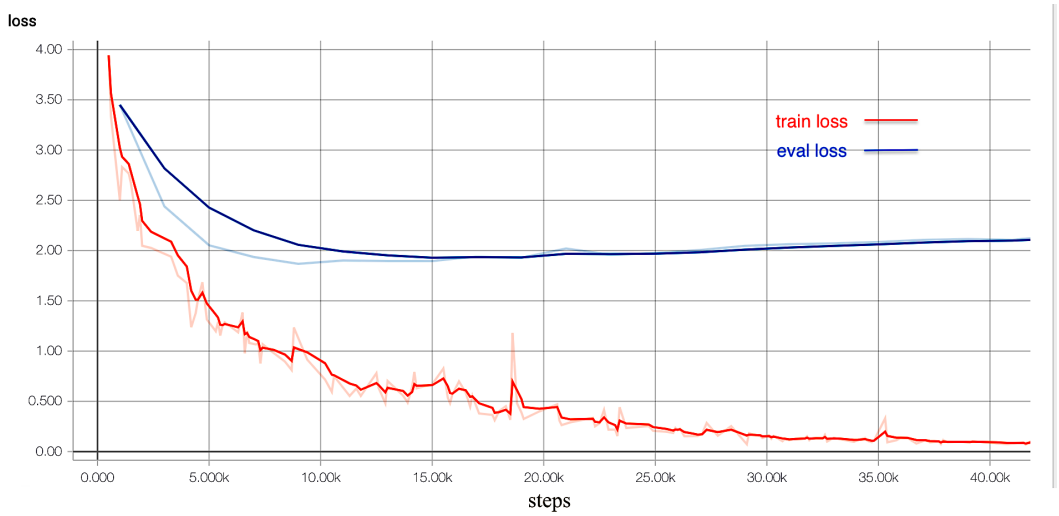
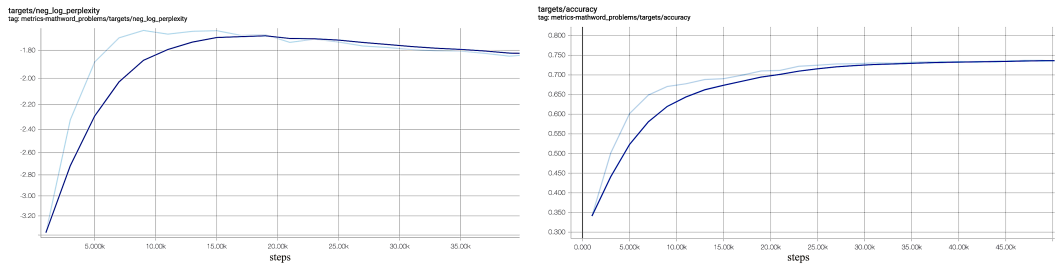


Figure 4: Training and Validation Loss for transformer models. This graph indicates that we should early stop at 15000 steps.



(a) Negative Log Perplexity

(b) Accuracy

Figure 5: Negative Log Perplexity and Accuracy of validation set during training for transformer models.

metric	scores on the dev set at 15,000 steps
loss	1.896
accuracy	0.8476
negative log perplexity	-1.667

Table 4: Results from Transformer at early stopping time 15,000 steps

5 Analysis

```
X = 8 + 27 ; 12
X = 6 - 3 ; 4
X = <unk> + <unk> + <unk> ; <unk>
X = <unk> + <unk> ; <unk>
X = <unk> + <unk> ; 19.02
```

```
X = <unk> - 0.5 ; 0.5
X = 0.25 + 0.375 + 0.25 ; 54
X = 1.16 + 1.0 ; 14.8
X = <unk> - <unk> ; <unk>
X = 6 - 2 ; 4
```

Figure 6: Examples of outputs from baseline models

An example of the output of the baseline is shown in **Figure 6**. One of the improvement that the transformer achieves is that it never outputs any `<unk>` since it trains its own embeddings using the traning data.

The challenge of evaluating a MWPs solver's performance comes from the hard definition of an "accurate" mathematical equations. These are several reasons behind:

- Adding spacing between elements in the equation shouldn't affect its mathematical expressions.
- Additions and multiplications follow commutative laws. Therefore the ordering of the addends and factors doesn't matter.
- Variables' names make no difference in the meaning of mathematical equations. No matter you are solving x, y or t, the results should be same, even the expressions don't exactly match to each other.
- Number of significant digits does affect the results.

Therefore, we conducted qualitative error analysis here to evaluate the transformer model compared to the baseline performance in the following several cases. Some details are shown below.

- Q: Jason had Pokemon cards . He gave 9 to his friends. He now has 4 Pokemon cards . How many Pokemon cards did he have to start with ?
  Correct output: $X - 9 = 4$
  BiGLAtt output: $x - 9 = 4 - X = 18$
  BiLLAtt output: $x - 9 = 4$
  Transformer output: $X - 9 = 4$

- Q: Mary has 9 yellow marbles Joan has 3 yellow marbles . How many yellow marbles do they have in all ?
  Correct output: $X = 9 + 3$
  BiGLAtt output: $X = 2 + 9$
  BiLLAtt output: $9 + 3 = x$
  Transformer output: $9 + 3 = X$

- Q: At Lindsey 's Vacation Wear , 0.375 the garments are bikinis and 0.25 are trunks . What fraction of the garments are either bikinis or trunks?
  Correct output: $X = 0.375 + 0.25$
  BiGLAtt output: $X = 0.25 + 0.25$
  BiLLAtt output: $X = 0.75 - 0.5$
  Transformer output: $X = 0.375 + 0.25$

Transformer models performs well in other outputs in very complex situation, where the baseline model doesn't test on. E.g.

- Q: What is the fourth root of 400 over root 10?
  Correct: x = 400 ^ (0.25)/sqrt(10)
  Transformer Output: t = 400 ^ (1/4)/10 ^(1/2)
  though the performance may be bad if solely evaluated using accuracy.

However, for problems with 'word numbers' and 'numerals' together, equations are hard to be established and result is bad.

- Q: If one third of 3/4 of a number is 21. Find the number?
  Correct: $1/3 * 3/4 * x = 21$
  Transformer Output: $x = 72$

In addition, for problems which need common senses and knowledge outside the problem itself, transformer also has bad results. E.g.

- Q: How do you add/subtract degrees, minutes, and seconds? For example, 71°18'-47°29'
  Correct: $x = 71 * 60 + 18 - 47 * 60 + 29$
  Transformer Output: $x = (47 + 29)/60$

though it successfully predicts the conversion ratio 60, which is not appear in the original MWP.

Although quantitative metrics are not reliable, the performance for transformer is not enough satisfying. The model still has room to be improved and feature engineering may still be necessary to capture the information that is not explicitly stated in the problem text.

6 Conclusion

In this work, we explore the deep learning methods to solve MWPs problem, where the task is to output one or sets of equations, which solve a simple algebra mathematical problem in text. To this end, we collected and preprocessed more than 45,000 question-equation pairs, although some of them may not be perfectly cleaned. We applied two bidirectional RNNs and transformer models and used embeddings trained for existing datasets to this task. Experiments show that transformer outperforms RNN baselines, in both the ability to solve the problem and capability to output a valid equation and digits. In the future, more work could be done to combine feature engineering with sole deep learning tasks to boost the performance.

7 Additional Information

Mentor: Special thanks to Anand Dhoot, our CS224N staff mentor

References

[1] D. Bobrow. Natural language input for a computer problem solving. page 146–226, 1964.

[2] CS224N Teaching Staff. Assignment 4, 2019.

[3] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, , and Hannaneh Hajishirzi. 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*, page 1152–1157, 2016.

[4] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. Learning to automatically solve algebra word problems. *Association for Computational Linguistics*, pages 271–281, 2014.

[5] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1015. URL `http://aclweb.org/anthology/P17-1015`.

[6] Anirban Mukherjee and Utpal Garain. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29:93–122, 2018.

[7] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=H1gR5iR5FX`.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, pages 6000–6010, 2017.

[9] Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018. URL `http://arxiv.org/abs/1803.07416`.

[10] Yan Wang, Xiaojiang Liu, and Shuming Shi. Deep neural solver for math word problems. *EMNLP*, 2017.

[11] Dongxiang Zhang, Lei Wang, Nu Xu, Bing Tian Dai, and Heng Tao Shen. The gap of semantic parsing: A survey on automatic math word problem solvers. *arXiv:1808.07290*, 2018.

8 Appendix

8.1 An example of data from MaWPS

Below is an example of data from MaWPS:


```
{
    "iIndex": 3903,
    "lAlignments": [
        15, 45],
    "lEquations": [
        "X=(900.0-156.0)"],
    "lSolutions": [
        744.0],
    "sQuestion": " Isabel bought 900 pieces of paper. She used 156 pieces

    of the paper. How many pieces of paper does she have left?"
}
```


Figure 7: Example of data from MaWPS

8.2 An example of data from Dolphin 18K

Below is an example of data from Dolphin 18K:


```
{
  "original_text": "",
  "sources": [
    "https://answers.yahoo.com/question/index?qid=20091207064212AAKlFPn"
  ],
  "flag": 0,
  "ans": "{9; 14}",
  "equations": "unkn: x,y\r\nequ: x + y = 23\r\nequ: x = 5 + y",
  "id": "yahoo.answers.20091207064212aaklfpn"
}
```


Figure 8: Example of data from Dolphin 18K

8.3 An example of data from AQUA-RAT

Below is an example of data from AQUA-RAT:


```
{
"question": "A grocery sells a bag of ice for 1.25, and makes 20\% profit.
                    If it sells 500 bags of ice, how much total profit does it make?",
"options": ["A)125", "B)150", "C)225", "D)250", "E)275"],
"rationale": "Profit per bag = 1.25 * 0.20 = 0.25
                    Total profit = 500 * 0.25 = 125
                Answer is A.",
"correct": "A"
}
```


Figure 9: Example of data from AQUA-RAT