QUESTION ANSWERING OVER FINANCIAL DATA

Aparimeya Taneja & Zihao lin
Duke University
{aparimeya.taneja,brain}@duke.edu

ABSTRACT

NLP solutions to numerical reasoning are critical in the domain of analysing financial documents. However, current models' results are not satisfying because they often miss the inherent properties of numbers themselves and the influence of numbers in the text. In this paper, we propose a **N**umber-**A**ware **E**ncoder-**D**ecoder Model (NAED) which introduces two modules to the baseline model proposed by Chen et al. (2021). First, we add a digit-to-digit encoder to measure the inherent properties of numbers themselves. Second, we implement number-aware attention to handle the influence of numbers to the input text. After adding the two modules into the FinQA baseline, we conduct experiments on the FinQA dataset to verify the effectiveness of NAED. The results with the model yield an improvement over the baseline. Our best model performed with an execution accuracy score of 0.68, and a program accuracy score of 0.67. Thus overall, we see an accuracy increase of 1.2%. Key challenges, namely training resources and multi step formulas, were also identified. Our code can be found at our repo here.

1 INTRODUCTION

Analyzing financial statements is an important task for people who work in finance, and most often this is how a company's business performance is evaluated. However, financial statements contain vast amounts of unstructured information which makes them difficult to analyze and doing calculations from this kind of data is a time-consuming task. Further, substantial knowledge/expertise about finance is a prerequisite to being able to ask meaningful financial questions, which involve complex and multi step numerical reasoning to answer. Thus, most general-domain QA models are not easily adaptable to answer realistic, complex financial questions. For example, the DROP dataset (Dua et al., 2019a) focused on Wikipedia-based questions that require numerical reasoning, e.g., "Where did Charles travel to first, Castile or Barcelona?" only needs a comparison between the times of two events (one-step calculation). This task is meaningful to the finance and research communities, and would help financial analysts in their work of parsing through financial statements. Our paper aims to further progress NLP solutions in this domain.

To handle these added complexities in our specific domain, Chen et al. (2021) proposed an encoder-decoder model as well as a novel expert annotated dataset: FinQA. This model however, does not reach a performance high enough to be used in practice. The model consists of two parts: a retriever and a generator. The retriever is a classifier based on BERT Devlin et al. (2018), and has the goal to retrieve the supporting facts from the input corpus. Whereas, the generator aims to generate the executable math expression(s). This is very basic and still has many remaining problems that need to be solved. Firstly, to retrieve useful sentences from hybrid data of table and text, the retriever needs to change each row of tables into a sentence by a certain format which causes many identical sentences. However, too many identical sentences in format can harm the ability of the retriever. Second, the generator does not measure the inherent properties of numbers themselves which may be very helpful to this task requiring quantitative analysis of numbers. Third, the generator does not consider the relationship between words and numbers and also the relationship between each pair of numbers. For example, the word ”year” is more related to the number ”2015” than the number ”10000” which is followed by the word ”dollars”.

In order to better focus the scope of our research on solving the accuracy of our model in answer quantitative questions which involve numerical reasoning over financial data, we essentially over-write the use of the retriever. Instead of doing the time consuming task of training the retriever, we give expert annotated 'gold' facts and statements directly to our generator[1] that come with the FinQA dataset. Therefore, NAED only has a generator (but does not have a retriever), and the input text is the 'gold' facts (which are used as labels for the retriever in the baseline model) extracted from the data source. To address our second issue about the properties of numbers, we add a digit-to-digit number encoder proposed by Wu et al. (2021) that explicitly incorporates inherent properties of numbers themselves. For the third issue, we use an implementation of attention mechanism to capture the relationship between numbers and words, as well as pairs of numbers simultaneously.

The main contributions of our project is summarized as follows:

- We add a digit-to-digit encoder module to the baseline model. This module improves the ability to measure the inherent properties of numbers themselves.
- We design a number-aware attention mechanism to measure the relationship between words, numbers and pairs of numbers.

2 Related Work

2.1 Question Answering

There exist several QA datasets involving numerical understandings and calculations. A majority of these sources are in the form of structured tables or knowledge bases. Popular datasets include ComplexWebQuestions (Talmor & Berant, 2018), WikiTableQuestions (Pasupat & Liang, 2015), Spider (Yu et al., 2018), etc.

The dataset most related to FinQA's dataset is the DROP dataset (Dua et al., 2019a), which applies simple calculations over texts. The top methods in DROP typically use specific prediction heads for each kind of calculation. HybridQA (Chen et al., 2020b) targets QA over both the table and the text, but not with the focus of numerical reasoning. All these datasets are built upon the general domain, mostly based on Wikipedia, whereas the FinQA dataset focuses on the finance domain, covering much more complex numerical reasoning questions, combining both structured tables and unstructured texts.

2.2 Financial QA

The current literature in this domain is still not accurate enough to be used in the field. Firstly, we looked at FinQA (Chen et al., 2021). This paper's novelty is its accompanying dataset: a large set of question-answer pairs over Financial reports, written by financial experts. Further, their model implements step memory as a mechanism for trying to solve multi-step questions.

Another interesting paper (Zhu et al., 2021) extracts samples from real financial reports to build a new large-scale QA dataset containing both Tabular And Textual data, named TAT-QA. TAT-QA proposes a novel QA model TAGOP, which is capable of reasoning over both tables and text. It adopts sequence tagging to extract relevant cells from the table along with relevant spans from the text to infer their semantics, and then applies symbolic reasoning over them with a set of aggregation operators to arrive at the final answer.

2.3 Numerical Reasoning

Some current studies use numerically-aware graph neural networks (GNNs) to analyze numerical reasoning over numbers. Ran et al. (2019) proposed NumNet which utilizes a GNN to measure the comparing information between numbers. The QDGAT model, proposed by Chen et al. (2020a), utilizes a graph neural network to represent the relationship between numbers or numbers and entities, which achieves the SOTA result on DROP dataset Dua et al. (2019b). Some approaches include modifying existing models such as Andor et al. (2019), where a modified BERT reading comprehension model was pretrained for numerical reasoning. Another approach is explicitly measuring number information Wu et al. (2021; 2020). The NumS2T model proposed by Wu et al. (2021)

[1]'Gold' here means facts and statements manually selected from a document that contain relevant information and numerical values that are needed to answer the given question.

introduces digit-to-digit number encoding to measure the properties of numbers, along with multi-layer perceptrons to classify the categories of numbers. These MLPs are trained by measuring the contribution of each number to the final answer. The NumS2T performs best on the Math23K Wang et al. (2017) and APE Zhao et al. (2020) datasets.

3 Approach

Figure 1 shows an overview of our proposed NAED model. We first set up notations in Section 3.1 and then Section 3.2 describes the details of our model.

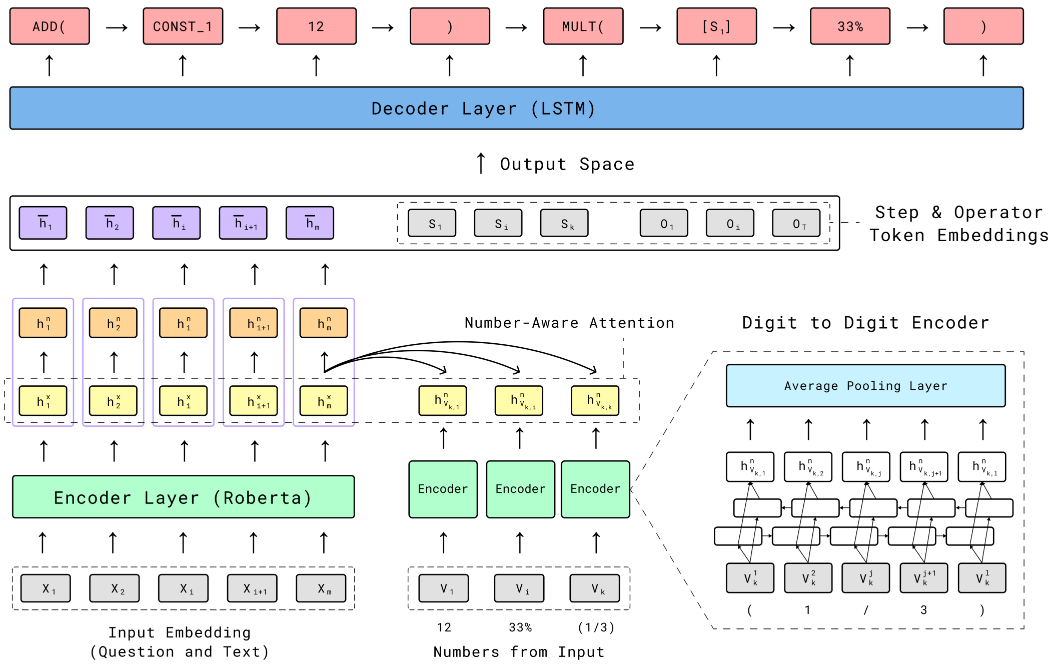


Figure 1: Model Architecture

3.1 Problem Definition

The input words embedding, containing questions and text, is denoted as X $= (x_1, x_2, \ldots, x_m)$. Our goal is to generate a sequence of token representing executable math expression Y $= (y_1, y_2, \ldots, y_n)$.

Here, we use additional notations to denote the numbers from the input data on their order of appearance. Let $\mathrm{V_c} = (v_1, v_2, \ldots, v_K)$ be the K numbers in the input text. The sequence of l characters of the k-th number is denoted as $\left(v_k^1, v_k^2, \ldots, v_k^l\right)$.

The generated token Y comes from three parts: 1) the input word X. 2) The step token S $= (s_1, s_2, \ldots, s_G)$ represents the results from previous steps, e.g. #0, #1, etc. 3) Operators tokens O $= (o_1, o_2, \ldots, o_T)$ contains operators, e.g. add, divide, etc and also some common constants, e.g. CONST_100, etc. Note that X, S, and O represent token embeddings.

3.2 Method

Following the baseline of FinQA Chen et al. (2021), which is an encoder-decoder model, we first use pre-trained models to encode input text X and denote the output as $\mathrm{h^x} = (h_1^x, h_2^x, \ldots, h_m^x)$.

For evaluating our model, rather than directly feed hidden states h^{x} into the decoder, we add two modules to solve the issues mentioned in the section 1 into NAED: digit-to-digit encoder and number-aware attention.

3.2.1 Digit-to-digit Encoder

In order to solve the issue that the generator does not measure the inherent properties of numbers themselves, we need to incorporate explicit information for each number. However, there are more than millions of different numbers that could appear in the input data. Therefore, inspired by Wu et al. (2021), we incorporate a digit-to-digit encoder into the FinQA base model. For example, for a 5-digit value $v_k = (1/3)$, we have $v_k^1 = ($, $v_k^2 = 1$, $v_k^3 = /$, $v_k^4 = 3$ and $v_k^5 =)$. After feeding the v_k into a char embedding layer $\mathrm{E_{char}}$, we have $\mathrm{E_{char}}(v_k) \in \mathrm{R}^{5\times \mathrm{d_{emb}}}$, where $\mathrm{d_{emb}}$ is the dimension of embeddings. Then, we feed the embeddings into a BiLSTM network followed by an average pooling layer to obtain the number of hidden states $\mathbf{h_{v_k}}$:

$$\mathbf{h^n_{v_{k,j}}} = \mathrm{BiLSTM}\left(\mathrm{E_{char}}\left(v_k^j\right), \mathbf{h^n_{v_{k,j-1}}}\right), \tag{1}$$

$$\mathbf{h^n_{v_k}} = \frac{1}{l}\sum_{j=1}^{l} \mathbf{h^n_{v_{k,j}}} \tag{2}$$

3.2.2 Number-aware Attention

To solved the third issue metioned in the Section 1, we need to capture the relations and dependencies between numeral pairs, as well as words and numbers. We can do so simultaneously, given that the numbers and words both form the whole input corpus.

We use an attention mechanism Bahdanau et al. (2014) on the output hidden embedding h^{x}. However, before calculating the attention score, we first replace the $\mathrm{h}^{\mathrm{x}}_{\mathbf{v_k}}$ of numbers in the input data by the output of the digit-to-digit encoder which is denoted as $\mathbf{h^n_{v_k}}$. By doing so, we can use only one attention layer to capture both the relations between number pairs (self-attention) and words and numbers.

$$\alpha_i = \mathrm{softmax}\left(\left(\mathbf{H^n_v}\right) \mathrm{W}_{\mathbf{h}} \mathbf{h}_i^{\mathbf{x}}\right), \tag{3}$$

$$\mathbf{h}_i^{\mathbf{n}} = \alpha_i \cdot \mathbf{H^n_v} \tag{4}$$

where α_i is the attention distribution of token i of all the input data. Combining $\mathbf{h^n}$ and $\mathbf{h^x}$, feeding it into a linear layer, we get $\overline{\mathbf{h}}$ which consists of both the information of encoder and the attention layer. We concatenate $\overline{\mathbf{h}}$, step tokens $\mathbf{S}$ and operator tokens $\mathbf{O}$ together as the input of our decoder.

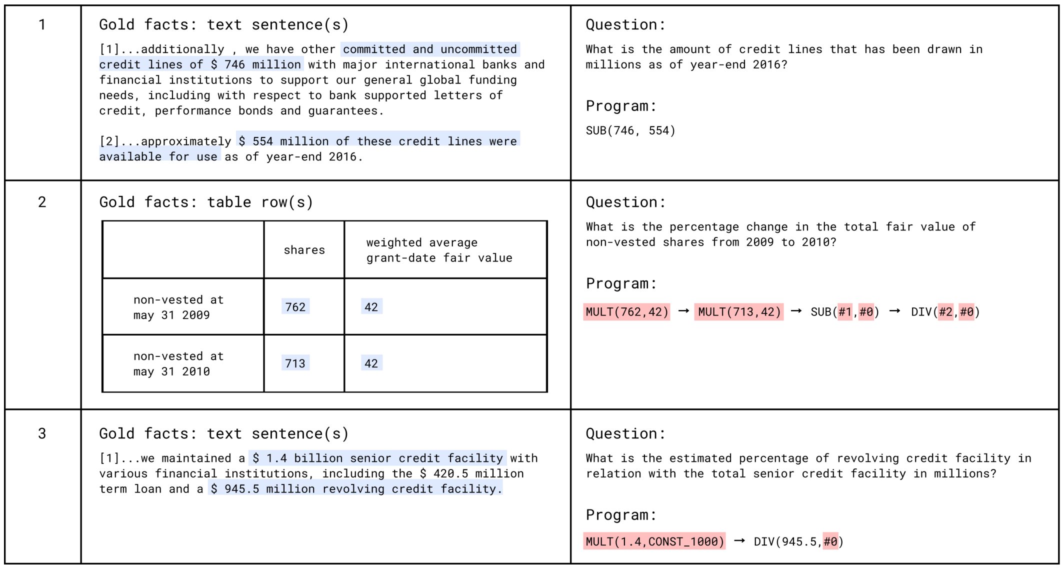
3.2.3 Decoder

We first followed the baseline of FinQA model exactly as described in the paper Chen et al. (2021). We use a BiLSTM as the decoder. During generating the sequence of expressions, every fourth step, when we generate the right brackets), we will update the step memory in the same way as the baseline Chen et al. (2021). We ignore the details of step memory module because that is not related to our contribution.

4 Experiments

4.1 Datasets and Evaluation

Previous studies on QA with numerical reasoning only evaluate the execution accuracy, i.e., the final results from the generated programs, such as DROP (Dua et al., 2019a) and MathQA (Amini et al., 2019). However, the applications for the finance domain generally pose much higher requirements of explainability and transparency. Here, a distinct advantage FinQA has is their inclusion of gold



1

Gold facts: text sentence(s)

[1]...additionally , we have other committed and uncommitted credit lines of $ 746 million with major international banks and financial institutions to support our general global funding needs, including with respect to bank supported letters of credit, performance bonds and guarantees.

[2]...approximately $ 554 million of these credit lines were available for use as of year-end 2016.

Question:
What is the amount of credit lines that has been drawn in millions as of year-end 2016?

Program:
SUB(746, 554)

2

Gold facts: table row(s)

	shares	weighted average grant-date fair value
non-vested at may 31 2009	762	42
non-vested at may 31 2010	713	42

Question:
What is the percentage change in the total fair value of non-vested shares from 2009 to 2010?

Program:
MULT(762,42) → MULT(713,42) → SUB(#1,#0) → DIV(#2,#0)

3

Gold facts: text sentence(s)

[1]...we maintained a $ 1.4 billion senior credit facility with various financial institutions, including the $ 420.5 million term loan and a $ 945.5 million revolving credit facility.

Question:
What is the estimated percentage of revolving credit facility in relation with the total senior credit facility in millions?

Program:
MULT(1.4,CONST_1000) → DIV(945.5,#0)

Figure 2: Three examples from the FinQA dataset. In [1] and [3], gold facts in the form of text is given to the generator, whereas table data is used in [2]. The blue highlighted sections in the input contain numerical values that need to be read in by the NAED Model to answer the given question correctly. The red highlighted sections in the output correspond to step memory. Order of operations in the output is also important here, signified by arrows.

programs for the dataset. This allows us to also test for program accuracy. Specifically, this is done by replacing all the arguments in a program with symbols, and then evaluating if two symbolic programs are mathematically equivalent. For example, the following two programs are equivalent programs:

$$\text{add}(a_1, a_2), \text{add}(a_3, a_4), \text{subtract}(\#0, \#1)$$
$$\text{add}(a_4, a_3), \text{add}(a_1, a_2), \text{subtract}(\#1, \#0)$$

Note that execution accuracy tends to overestimate the performance because sometimes the model just hit the correct answer by chance; While program accuracy tends to produce false negatives since some questions may have multiple correct programs Chen et al. (2021).

4.2 Model and Training Details

In this paper, most hyperparameters are set followed by the paper Chen et al. (2021). We truncate the input text to a max sequence length of 512, the output math expression to a max sequence length of 30, and the sequence of characters of numbers to a max sequence length of 10. The embedding dimension $\mathrm{d_{emb}}$ is set to 300. If there is one character that is unknown, the character will be set UNK. The hidden size is 768, the dropout Srivastava et al. (2014) is 0.1 and learning rate is 2e-5. Step tokens and operators tokens have totally 44 tokens. We use one A100 GPU to run our experiments, while due to the limit of GPU resource, we set batch size to 12 which is different from the baseline batch size 16. Therefore, the final performance of the baseline model run by ourselves is lower that that shown in the original paper Chen et al. (2021). We set training epochs to 300.

Followed the best model architecture, we use RoBERTa-large Liu et al. (2019) as the encoder of input text and BiLSTM as the decoder.

4.3 Results

We trained the baseline of FinQA by ourselves first and then train our NAED model to compare the results. Table 1 shows the results of experiments.

Due to the lower batch size, the FinQA baseline model run by ourselves gives us lower accuracy than that of original model shown in the paper where the batch size is 16. The execution accuracy of the baseline we run is 66.96% while that of the baseline of original paper is 70%. The program accuracy of the baseline we run is also lower than that of the original baseline.

The third row of Table 1 shows that the execution accuracy of NAED is 68.18 % which is higher than the baseline we run, and the program accuracy is 67.13% which is also higher than that of baseline we run, 65.82%. Note that we also set the batch size for NAED to 12. This shows that our model indeed improves the performance of the baseline. The modules we add is efficient.

However we can see from the last row of the Table 1 that our model is still far behind the human performance. We still need more new techniques on this task.

Model	Exec. Acc. #1	Prog. Acc. #2
FinQA (Gold Generator, Original)	0.7000	0.6876
FinQA (Gold Generator, Run-by-ourselves)	0.6696	0.6582
NAED	0.6818	0.6713
General Crowd	0.5068	0.4817
Human Expert Baseline	0.9116	0.8749

Table 1: Results across models

5 Analysis

We compare the number of correct predicted of NAED and the baseline model in this section and analyze the results. Here, correct means the execution expression correct but not the final result correct.

5.1 Different Data Types

We exam how NAED model performs on different types of input data: text only, table only and text-table, which means the input data is only from text, only from table and from both text and table separately. We can see from the table 2 that, for each types of data, the NAED model has more correct items than the baseline we run. There is no difference of the improvement of the performance. That makes sense because the two modules we add are aimed to extract the inherent properties of numbers. Therefore, whether the number comes from text or table, the model can handle either.

5.2 Different Steps

We also test the questions which require varying steps: 1 step, 2 steps and $n(> 2)$ steps. From the table 2, we can see that the NAED has 502 correct predictions which is 17 higher than that of the baseline we run. This means that these two modules significantly improve the ability to understand numbers if we only require one step calculation. However, for the questions that require 2 steps to answer, the performances of NAED and the baseline we run are very close, which shows that the efficiency of the two modules is not obvious. Further, the performance of NAED is worse than the baseline we run for the questions that require more than 2 steps.

5.3 Require Const or Not

We also exam the questions require constant or not. We found that the performance of NAED for those which do not require constant is better than the baseline we run, while is worse for those ehich require.

The reason that NAED performs worse than the baseline we run on the multi-step questions and constant-required questions is same. For multi-step questions, the final predictions need to contain step tokens s such as #1, #2, etc; and for constant-required questions, the final predictions require some constant tokens (denotes as a part of operator tokens) o, such as CONST_100. From the figure 2, we can see that the added two modules help to enhance the encoder for only the input data, and after we get h, we concatenate it with step tokens S and operator tokens O. That results in that, in input of the decoder, the part from the input data contains much more useful information than step and operator tokens, which makes the decoder put more attention on the inout data but sometimes ignore the step tokens and operator tokens that also required to the answers. This is one of the shortcomings for our model.

To sum up, the two added modules indeed improve the model's ability to understand the numbers but harm the ability to answer questions that require additional tokens.

Methods	C/W	NAED	Baseline We Run
Performances on table and text			
text-only questions	correct	169	163
	wrong	114	120
table-only questions	correct	533	528
	wrong	173	178
text-table questions	correct	80	77
	wrong	78	81
Performances on different steps			
1-step questions	correct	502	485
	wrong	155	172
2-steps questions	correct	260	259
	wrong	146	147
n($>$ 2)-step questions	correct	20	24
	wrong	64	60
Requires constant or not			
requires constant	correct	58	62
	wrong	110	106
not requires constant	correct	724	706
	wrong	255	273

Table 2: Results for different types of questions

6 Conclusion

In this paper, we propose a new model NAED which adds two modules to the baseline model: digit-to-digit number encoding and number-aware attention. These capture the inherent properties and the relations between number pairs as well as between numbers and words. In doing so, we successfully improve the accuracy from the baseline we run. However, this model has one shortcoming, that although there is an improvement to the overall performance, problems that require additional tokens (as are needed in multi-step reasoning as well as constant tokens) perform worse.

The goal for future work would involve finding new and better techniques to capture the information of numbers. For example, we can take the additional tokens into the encoder first rather than directly concatenate their embeddings. Further, we need to find some ways to improve the retriever or just replace it by some other mechanism. Apart from these fundamental issues, the largest challenges here were certainly computing resource and data scarcity.

Author Contributions

Both Aparimeya and Zihao worked on the proposal. AP recreated FinQA for initial findings for the presentation. The presentation itself was made and presented by both AP and Zihao. Zihao then ran and modified FinQA for the final project. Zihao laid out the NAED model, and its diagram was illustrated by AP. Both AP and Zihao worked on writing different parts of the final paper (AP: Abstract and sections 1, 2, 4.1, 6. Zihao: sections 3, 4.2, 4.3, 5 , 6). Examples were illustrated by AP.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 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, Volume 1 (Long and Short Papers)*, pp. 2357–2367, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1245. URL https://aclanthology.org/N19-1245.

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. Giving bert a calculator: Finding operations and arguments with reading comprehension, 2019.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Kunlong Chen, Weidi Xu, Xingyi Cheng, Zou Xiaochuan, Yuyu Zhang, Le Song, Taifeng Wang, Yuan Qi, and Wei Chu. Question directed graph attention network for numerical reasoning over text, 2020a.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1026–1036, Online, November 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.91. URL https://aclanthology.org/2020.findings-emnlp.91.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. Finqa: A dataset of numerical reasoning over financial data, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, Minneapolis, Minnesota, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019b.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1470–1480, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1142. URL https://aclanthology.org/P15-1142.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. Numnet: Machine reading comprehension with numerical reasoning, 2019.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 641–651, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1059. URL https://aclanthology.org/N18-1059.

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

Qinzhuo Wu, Qi Zhang, Jinlan Fu, and Xuan-Jing Huang. A knowledge-aware sequence-to-tree network for math word problem solving. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7137–7146, 2020.

Qinzhuo Wu, Qi Zhang, Zhongyu Wei, and Xuan-Jing Huang. Math word problem solving with explicit numerical values. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 5859–5869, 2021.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3911–3921, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1425. URL https://aclanthology.org/D18-1425.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. Ape210k: A large-scale and template-rich dataset of math word problems. *arXiv preprint arXiv:2009.11506*, 2020.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. *arXiv preprint arXiv:2105.07624*, 2021.