Giving BERT a Calculator: Finding Operations and Arguments with Reading Comprehension

Daniel Andor, Luheng He, Kenton Lee, Emily Pitler
Google Research
{andor, luheng, kentonl, epitler}@google.com

Abstract

Reading comprehension models have been successfully applied to extractive text answers, but it is unclear how best to generalize these models to abstractive numerical answers. We enable a BERT-based reading comprehension model to perform lightweight numerical reasoning. We augment the model with a predefined set of executable 'programs' which encompass simple arithmetic as well as extraction. Rather than having to learn to manipulate numbers directly, the model can pick a program and execute it. On the recent Discrete Reasoning Over Passages (DROP) dataset, designed to challenge reading comprehension models, we show a 33% absolute improvement by adding shallow programs. The model can learn to predict new operations when appropriate in a math word problem setting (Roy and Roth, 2015) with very few training examples.

1 Introduction

End-to-end reading comprehension models have been increasingly successful at extractive question answering. For example, performance on the SQuAD 2.0 (Rajpurkar et al., 2018) benchmark has improved from 66.3 F1 to 89.5[1] in a single year. However, the Discrete Reasoning Over Passages (DROP) (Dua et al., 2019) dataset demonstrates that as long as there is quantitative reasoning involved, there are plenty of relatively straightforward questions that current extractive QA systems find difficult to answer. Other recent work has shown that even state-of-the-art neural models struggle with numerical operations and quantitative reasoning when trained in an end-to-end manner (Saxton et al., 2019; Ravichander et al., 2019). In other words, even BERT (Devlin et al., 2019) is not very good at doing simple calculations.

[1] https://rajpurkar.github.io/SQuAD-explorer/

How many more Chinese nationals are there than European nationals?
The city of Bangkok has a population of 8,280,925 ...the census showed that it is home to 81,570 Japanese and **55,893** Chinese nationals, as well as 117,071 expatriates from other Asian countries, **48,341** from Europe, 23,418 from the Americas,...
NAQANet: −55893 **Ours**: `Diff`(55893, 48341) = **7552**

Table 1: Example from the DROP development set. The correct answer is not explicitly stated in the passage and instead must be computed. The NAQANet model[2](Dua et al., 2019) predicts a negative number of people, whereas our model predicts that an operation `Diff` should be taken and identifies the two arguments.

In this work, we extend an extractive QA system with numerical reasoning abilities. We do so by asking the neural network to synthesize small programs that can be executed. The model picks among simple programs of the form `Operation`(args, ...), where the possible operations include span extraction, answering yes or no, and arithmetic. For math operations, the arguments are pointers to numbers in the text and, in the case of composition, other operations. In this way, the burden of actually doing the computation is offloaded from the neural network to a calculator tool. The program additionally provides a thin layer of interpretability that mirrors some of the reasoning required for the answer. For example, in Table 1, the model predicts subtraction (`Diff`) over two numbers in the passage, and executes it to produce the final answer.

We start with a simple extractive question answering model based on BERT (Devlin et al., 2019), and show the following:

1. Predicting unary and binary math operations

[2] https://demo.allennlp.org/reading-comprehension/NzQwNjg1

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 5947–5952, Hong Kong, China, November 3–7, 2019.

with arguments resulted in significant improvements on the DROP dataset.

2. Our model can smoothly handle more traditional reading comprehension inputs as well as math problems with new operations. Co-training with the CoQA (Reddy et al., 2018) dataset improved performance on DROP. The DROP+CoQA trained model had never seen multiplication or division examples, but can learn to predict these two ops when appropriate in a math word problem setting (Roy and Roth, 2015) with very few training examples.

2 Background and Related Work

Discrete Reasoning over Paragraphs (DROP) (Dua et al., 2019) is a reading comprehension task that requires discrete reasoning. Inspired by semantic parsing tasks where models need to produce executable 'programs', it keeps the open-domain nature of reading comprehension tasks such as SQuAD 2.0 (Rajpurkar et al., 2018). As shown in Table 1, the system needs to perform fuzzy matching between *"from Europe"* and *"European nationals"* in order to identify the arguments.

Numerically-aware QANet (NAQANet) (Dua et al., 2019) is the current state-of-the-art[3] system for DROP. It extends the QANet model (Yu et al., 2018) with predictions for numbers (0–9) and summation operations. For the latter, it performs a 3-way classification (plus, minus, and zero) on all the numbers in the passage.

While certain binary operations are expressible efficiently with flat sign prediction, it is difficult to generalize the architecture. Moreover, each number is tagged independently, which can cause global inconsistencies; for instance, in Table 1 it assigns a single minus label and no plus labels, leading to a prediction of negative people.

Mathematical Word Problems have been addressed with a wide variety of datasets and approaches; see Zhang et al. (2018) for an overview. One such dataset of arithmetic problems is the Illinois dataset (Roy and Roth, 2015). The problems are posed in simple natural language that has a specific, narrow domain, For example: *"If there are 7 bottle caps in a box and Linda puts 7 more bottle caps inside, how many bottle caps are in the box?"*. Unlike DROP, the problems are typically 1–3 sentences long and do not require reading complex passages. Instead, the main challenge is mathematical reasoning. According to Zhang et al. (2018), the current state of the art uses syntactic parses and deterministic rules to convert the input to logical forms (Liang et al., 2016).

3 Model

We extend a BERT-based extractive reading comprehension model with a lightweight extraction and composition layer. For details of the BERT architecture see Devlin et al. (2019). We only rely on the representation of individual tokens that are jointly conditioned on the given question Q and passage P. Our model predicts an answer by selecting the top-scoring *derivation* (i.e. program) and executing it.

Derivations We define the space of possible derivations $\mathcal{D}$ as follows:

- *Literals*: $\{\text{YES}, \text{NO}, \text{UNKNOWN}, 0, \ldots 9\}$.
- *Numerical operations*: including various types of numerical compositions of numbers[4], such as `Sum` or `Diff`.
- *Text spans*: composition of tokens into text spans up to a pre-specified length.
- *Composition of compositions*: we only consider two-step compositions, including merging text spans and nested summations.

The full set of operations are listed in Table 2. For example, `Sum` is a numerical operation that adds two numbers and produces a new number. While we could recursively search for compositions with deep derivations, here we are guided by what is required in the DROP data and simplify inference by heavily restricting multi-step composition. Specifically, spans can be composed into a pair of merged spans (`Merge`), and the sum of two numbers (`Sum`) can subsequently be summed with a third (`Sum3`). The results in Table 3 show the dev set oracle performance using these shallow derivations, by answer type.

Representation and Scoring For each derivation $d \in \mathcal{D}$, we compute a vector representation $\mathbf{h}_d$ and a scalar score $\rho(d, P, Q)$ using the BERT output vectors. The scores ρ are used for computing the probability $P(d \mid P, Q)$ as well as for pruning. For brevity, we will drop the dependence on P and Q in this section.

[3] https://leaderboard.allenai.org/drop/submissions/public

[4] Numbers are heuristically extracted from the text.

	Derivations	Example Question	Answer Derivation
Literals	YES, NO, UNKNOWN, 0, 1 ..., 9	How many field goals did Stover kick?	4
Numerical	`Diff100` : $n_0 \rightarrow 100 - n_1$	How many percent of the national population does not live in Bangkok?	$100 - 12.6 = 87.4$
	`Sum` : $n_0, n_1 \rightarrow n_0 + n_1$ as well as: `Diff`, `Mul`, `Div`	How many from the census were in Ungheni and Cahul?	$32,828 + 28,763 = 61591$
Text spans	`Span` : $i, j \rightarrow s$	Does Bangkok have more Japanese or Chinese nationals?	"Japanese"
Compositions	`Merge` : $s_0, s_1 \rightarrow \{s_0, s_1\}$	What languages are spoken by more than 1%, but fewer than 2% of Richmond's residents?	"Hmong-Mien languages", "Laotian"
	`Sum3` : $n_0, n_1, n_2 \rightarrow (n_0 + n_1) + n_2$	How many residents, in terms of percentage, speak either English, Spanish, or Tagalog?	$\texttt{Sum}(64.56, 23.13) + 2.11 = 89.8$

Table 2: Operations supported by the model. s, n refer to arguments of type *span* and *number*, respectively. i, j are the start and end indices of span s. The omitted definitions of `Diff`, `Mul`, and `Div` are analogous to `Sum`.

Literals are scored as $\rho(d) = \mathbf{w}_d^\intercal \text{MLP}_{\text{lit}}(\mathbf{h}_{\texttt{CLS}})$, where $\mathbf{h}_{\texttt{CLS}}$ is the output vector at the `[CLS]` token of the BERT model (Devlin et al., 2019).

Numeric operations use the vector representations $\mathbf{h}_i$ of the first token of each numeric argument. Binary operations are represented as

$$\mathbf{h}_d = \text{MLP}_{\text{binary}}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{h}_i \circ \mathbf{h}_j) \quad (1)$$

and scored as $\rho(d) = \mathbf{w}_{op}^\intercal \mathbf{h}_d$, where $\mathbf{h}_d$ represents the binary arguments and *op* is the operation type. $\circ$ is the Hadamard product. Unary operations such as `Diff100` are scored as $\mathbf{w}_{op}^\intercal \text{MLP}_{\text{unary}}(\mathbf{h}_i)$.

Text spans are scored as if they were another binary operation taking as arguments the start and end indices i and j of the span (Lee et al., 2017):

$$\mathbf{h}_d = \text{MLP}_{\text{span}}(\mathbf{h}_i, \mathbf{h}_j) \quad (2)$$

and scored as $\rho(d) = \mathbf{w}_{\text{span}}^\intercal \mathbf{h}_d$.

Compositions of compositions are scored with the vector representations of its children. For example, the ternary `Sum3`, comprising a `Sum` and a number, is scored with $\mathbf{w}_{\texttt{Sum3}}^\intercal \text{MLP}_{\texttt{Sum3}}(\mathbf{h}_{d0}, \mathbf{h}_k)$, where $\mathbf{h}_{d0}$ corresponds to the representation from the first `Sum`, and $\mathbf{h}_k$ is the representation of the third number. The *composition* of two spans is scored as $\mathbf{w}_{\texttt{Merge}}^\intercal \text{MLP}_{\texttt{Merge}}(\mathbf{h}_{d0}, \mathbf{h}_{d1}, \mathbf{h}_{d0} \circ \mathbf{h}_{d1})$, where $\mathbf{h}_{d0}$ and $\mathbf{h}_{d1}$ are span representations from (2). The intuition for including $\mathbf{h}_{d0} \circ \mathbf{h}_{d1}$ is that it encodes span similarity, and spans with similar types are more likely to be merged.

This strategy differs from the NAQANet baseline in a few ways. One straightforward difference is that we use BERT as the base encoder rather than QANet. A more meaningful difference is that we model all derivations in the unified op scoring framework described above, which allows generalizing to new operations, whereas NAQANet would require more large-scale changes to go beyond addition and subtraction. Generalizing the model to new ops is a case of extending the derivations and scoring functions. In Section 4, we will show the impact of incrementally adding `Diff100`, `Sum3`, and `Merge`.

3.1 Training

We used exhaustive pre-computed oracle derivations $\mathcal{D}^*$ following Dua et al. (2019). We marginalized out all derivations d^* that lead to the answer[5] and minimized:

$$\mathcal{J}(P, Q, \mathcal{D}^*) = -\log \sum_{d^* \in \mathcal{D}^*} P(d^* \mid P, Q)$$

$$P(d \mid P, Q) = \frac{\exp \rho(d, P, Q)}{\sum_{d'} \exp \rho(d', P, Q)}$$

If no derivation lead to the gold answer ($\mathcal{D}^*$ is empty), we skipped the example.

Pruning During inference, the `Merge` and `Sum3` operations are composed from the results of `Span` and `Sum` operations, respectively. The space of possible results of `Merge` is quadratic in the number $|\mathcal{S}|$ of possible spans. With $|\mathcal{S}| \sim 10^4$, the complete set of `Merge` instances becomes overwhelming. Similarly, with $|\mathcal{N}| \sim 100$ numbers in each passage, there are millions of possible `Sum3` derivations. To do training and inference efficiently, we kept only the top 128 `Span` and `Sum` results when computing `Merge` and `Sum3`.[6]

[5] In practice we capped the number of derivations at 64, which covers 98.7% of the training examples.

[6] During training, the pruned arguments had recall of 80–90% after 1 epoch and plateaued at 95–98%.

	Oracle	Overall Dev		Overall Test		Date (1.6%)		Number (62%)		Span (32%)		Spans (4.4%)	
	Dev EM	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
NAQANet		46.75	50.39	44.24	47.77	32.0	39.6	44.9	45.0	58.2	64.8	0.0	27.3
Our basic[7]	*80.03*	66.50	69.91	-	-	57.0	65.1	65.8	66.1	78.0	82.6	0.0	35.7
+`Diff100`	*88.75*	75.52	78.82	-	-	53.6	61.3	80.3	80.5	78.4	82.8	0.0	35.8
+`Sum3`	*90.16*	76.70	80.06	-	-	58.0	64.6	81.9	82.1	78.9	83.4	0.0	36.0
+`Merge`	*93.01*	76.95	80.48	-	-	58.1	61.8	82.0	82.1	78.8	83.4	5.1	45.0
+CoQA	*93.01*	**78.00**	**81.56**	**76.93**	**80.47**	**59.5**	**66.4**	**83.0**	**83.2**	**79.8**	**84.2**	**5.8**	**46.8**
+Ensemble	*93.01*	**78.95**	**82.54**	**78.15**	**81.78**	**59.7**	**67.7**	**83.9**	**84.1**	**81.2**	**85.5**	5.4	46.5
Oracle		*93.01*				*71.6*		*94.5*		*95.8*		*60.5*	

Table 3: Accuracies on the DROP dev and test set in terms of exact match (EM) and token-level F1. The right-hand columns show the performance breakdown with different answer types on the development set. The largest improvements come from *Date*, *Number*, and *Spans* (answers with multiple spans). *Oracle* rows and columns indicate the performance that could be achieved by perfect selections of derivations. The ensemble used 6 models.

Spurious ambiguities Of the answers for which we could find at least one oracle derivation, 36% had two or more alternatives. During training, the model became effective at resolving many of these ambiguities. We monitored the entropy of $P(d^* \mid P, Q)$ for the ambiguous examples as training progressed. At the start, the entropy was 2.5 bits, which matches the average ambiguous oracle length of $\sim$ 6 alternatives. By the end of 4 epochs, the average entropy had dropped to < 0.2 bits, comparable to a typical certainty of 95–99% that one of the derivations is the correct one.

4 Experiments

Our main experiments pertain to DROP (Dua et al., 2019), using DROP and, optionally, CoQA (Reddy et al., 2018) data for training. Preprocessing and hyperparameter details are given in the supplementary material. In addition to full DROP results, we performed ablation experiments for the incremental addition of the `Diff100`, `Sum3`, and `Merge` operations, and finally the CoQA training data. We ran on the CoQA dev set, to show that the model co-trained on CoQA can still perform traditional reading comprehension. To investigate our model's ability to do symbolic reasoning at the other extreme, we performed few-shot learning experiments on the Illinois dataset of math problems (Roy and Roth, 2015).

4.1 DROP Results

As shown in Table 3, our model achieves over 50% relative improvement (over 33% absolute) over the previous state-of-the-art NAQANet system. The ablations indicate that the improvements due to the addition of extra ops (`Diff100`, `Sum3`, `Merge`) are roughly consistent with their proportion in the data. Specifically, the `Diff100` and `Sum3` derivations increase the oracle performance by 8.7% and 1.4% respectively, corresponding to model improvements of roughly 9% and 1.1%, respectively. Answers requiring two spans occur about 2.8% of the time, which is a 60.4% proportion of the *Spans* answer type. `Merge` only improves the *Spans* answer type by 9%, which we think is due to the significant 11:1 class imbalance between competing single and multiple spans. As a result, multiple spans are under-predicted, leaving considerable headroom there.

Pre-training on CoQA then fine-tuning on DROP lead to our best results on DROP, reported in Table 3. After fine-tuning on DROP, the model forgot how to do CoQA, with an overall F1 score of 52.2 on the CoQA dev set. If one prefers a model competent in both types of input, then the forgetting can be prevented by fine-tuning on both CoQA and DROP datasets simultaneously. This resulted in dev set F1 scores of 82.2 on CoQA and 81.1 on DROP. The CoQA performance is decent and compares well with the pre-trained model performance of 82.5. The 0.5% drop in DROP performance is likely attributable to the difference between pre-training versus fine-tuning on CoQA.

We ensembled 6 models (3 seeds $\times$ 2 learning rates) for an additional 1% improvement.

4.2 Results on Math Word Problems

We trained our model on the Illinois math word problems dataset (Roy and Roth, 2015), which contains answers requiring multiplication and

[7] The "basic" model includes all $\mathcal{D}_{\text{direct}}$, all $\mathcal{S}$, and the simple binary operations `Sum` and `Diff`.

Roy et al. (2015)	73.9
Liang et al. (2016)	**80.1**
Wang et al. (2018)	73.3
Our basic: IL data	48.6 ± 5.3
+ `Mul` and `Div`	74.0 ± 6.0
+ DROP data	$\mathbf{83.2 \pm 6.0}$

Table 4: Accuracy on the Illinois (IL) dataset[8] of 562 single-step word problems, using the five cross-validation folds of Roy and Roth (2015). Standard deviations were computed from the five folds. Roughly half the questions require the use of `Sum` and `Diff`, and half require `Mul` and `Div`.

division—operations not present in DROP—as well as addition and subtraction, in roughly equal proportion. Given the small ($N = 562$) dataset size, training and evaluation is done with five-fold cross-validation on a standardized set of splits. As shown in Table 4, when we added `Mul` and `Div` to our basic DROP operations, the model was able to learn to use them. Transferring from the DROP dataset further improved performance beyond that of Liang et al. (2016), a model specific to math word problems that uses rules over dependency trees. Compared to other more general systems, our model outperforms the deep reinforcement learning based approach of Wang et al. (2018).

5 Conclusions and Future Work

We proposed using BERT for reading comprehension combined with lightweight neural modules for computation in order to smoothly handle both traditional factoid question answering and questions requiring symbolic reasoning in a single unified model. On the DROP dataset, which includes a mix of reading comprehension and numerical reasoning, our model achieves a 33% absolute improvement over the previous best. The same model can also do standard reading comprehension on CoQA, and focused numerical reasoning on math word problems. We plan to generalize this model to more complex and compositional answers, with better searching and pruning strategies of the derivations.

Acknowledgements

We would like to thank Chris Alberti and Livio Baldini Soares for tremendously helpful discussions, and we are grateful to all members of the Google Research Language team.

[8] https://cogcomp.org/page/resource_view/98

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL*.

Kenton Lee, Tom Kwiatkowski, Ankur P. Parikh, and Dipanjan Das. 2017. Learning recurrent span representations for extractive question answering. *CoRR*, abs/1611.01436.

Chao-Chun Liang, Kuang-Yi Hsu, Chien-Tsung Huang, Chung-Min Li, Shen-Yu Miao, and Keh-Yih Su. 2016. A tag-based statistical english math word problem solver with understanding, reasoning and explanation. In *IJCAI*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you dont know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Abhilasha Ravichander, Aakanksha Naik, Carolyn Penstein Rosé, and Eduard H. Hovy. 2019. Equate: A benchmark evaluation framework for quantitative reasoning in natural language inference. *CoRR*, abs/1901.03735.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2018. CoQA: A conversational question answering challenge. *CoRR*, abs/1808.07042.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *EMNLP*.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. *CoRR*, abs/1904.01557.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018. MathDQN: Solving arithmetic word problems via deep reinforcement learning. In *AAAI*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. *Proceedings of ICLR*.

Dongxiang Zhang, Lei Wang, Nuo Xu, Bing Tian Dai, and Heng Tao Shen. 2018. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE transactions on pattern analysis and machine intelligence*.