Have You Seen That Number? Investigating Extrapolation in Question Answering Models

Jeonghwan Kim **Giwon Hong*** **Kyung-min Kim***
Junmo Kang* **Sung-Hyon Myaeng**
School of Computing, KAIST
Daejeon, Republic of Korea
{jeonghwankim123, giwon.hong, kimdarwin,
junmo.kang, myaeng}@kaist.ac.kr

Abstract

Numerical reasoning in machine reading comprehension (MRC) has shown drastic improvements over the past few years. While the previous models for numerical MRC are able to interpolate the learned numerical reasoning capabilities, it is not clear whether they can perform just as well on numbers unseen in the training dataset. Our work rigorously tests state-of-the-art models on DROP, a numerical MRC dataset, to see if they can handle passages that contain out-of-range numbers. One of the key findings is that the models fail to extrapolate to unseen numbers. Presenting numbers as digit-by-digit input to the model, we also propose the *E-digit* number form that alleviates the lack of extrapolation in models and reveals the need to treat numbers differently from regular words in the text. Our work provides a valuable insight into the numerical MRC models and the way to represent number forms in MRC.

1 Introduction

The research in question-answering (QA) models that are able to perform reading comprehension and discrete reasoning over numbers in the passage has seen significant progress, like the models in DROP (Ran et al., 2019; Hu et al., 2019; Chen et al., 2020; Geva et al., 2020). Despite their ability to understand the complex context and numbers within, none of them deal with the notion of whether these models can robustly handle "unseen" numbers during testing. The ability to extend discrete, symbolic rules such as addition and subtraction on numbers outside what we already know is called *extrapolation*, and this is an essential part of human intelligence. For example, if one can reason on numbers over text that range between 0 and 100, it is logically reasonable to infer that it should be able to handle numbers larger than 100. The lack of extrapolation capability in models, however, is a significant obstacle in the way toward a truly generalizable, number-understanding QA.

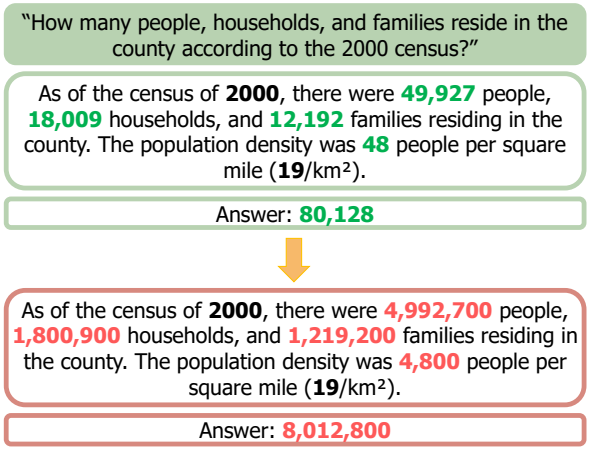

Figure 1: An example of DROP data instance perturbation to test the numerical QA models for extrapolation capabilities. This figure shows CARDINAL-type numbers factored by 100.

Although the problem of numerical extrapolation has been recently addressed by previous works in arithmetic word problem (AWP) settings (Trask et al., 2018; Madsen and Johansen, 2020; Kim et al., 2021) where the given instances involve simple math problems like *"What is 24 + 5?"*, their proposed approaches do not have the ability to handle two or more supporting facts (Kim et al., 2021), which is a capability demanded by DROP to handle multiple numbers, or deal with negative numbers or learn question-context relation (Trask et al., 2018). These limitations preclude the possibility of applying their extrapolation capability on the DROP task, where models are required to reason over multiple sentences while dealing with heterogeneous number types (e.g., percentage, cardinal, date), unlike in AWP settings where the numbers are simple, homogeneous type scalars. To see if the state-of-the-art models for DROP possess the extrapolation

*Equal contribution.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7031–7037
November 7–11, 2021.

capability we design the perturbated version of the DROP evaluation dataset as in Figure 1 (Section 3 for details). Surprisingly, the models show significant performance drop only when the range of numbers appearing in the passage is changed.

We also note that the models for DROP typically use transformer models as the encoder for context understanding. As shown in Wallace et al. (2019); Geva et al. (2020), subword tokenization methods arbitrarily subdivide the numbers and cause two very similar numbers to be in two disparate forms. This observation is in line with Nogueira et al. (2021)'s conclusion that how the numbers are presented to the model, or their *surface form*, influences the modeling of numbers. The surface forms proposed in Nogueira et al. (2021) provide digit-place information with a special set of tokens (the first three surface forms in Figure 2), to increase model's accuracy in a simple addition task. However, they fail to imbue the extrapolation capability in their tested models, observing that the addition rules could not be extended beyond the length of numbers seen during training. Therefore, we propose a new surface form called **E-digit** (Figure 2) that addresses the lack of extrapolation capability in the models. Our **E-digit** method successfully generalizes to out-of-distribution numbers and outperforms all the other surface forms by a significant margin.

2 Related Work

Previous works like NumNet (Ran et al., 2019) attempt to tackle the DROP task by using graphs to imbue the model with the relative magnitude information. GenBERT (Geva et al., 2020) pre-trains BERT (Devlin et al., 2019) with synthetic number and text data. QDGAT (Chen et al., 2020) designs a graph neural network with fully-connected number nodes of same entity type. While there are many other related works on this topic (Hu et al., 2019; Andor et al., 2019; Gupta et al., 2019; Min et al., 2019; Sundararaman et al., 2020; Saha et al., 2021), none of them address the problem of extrapolation in DROP. Although Wallace et al. (2019) reveals that NAQANet (Dua et al., 2019) struggles to deal with numbers outside the training range, showing a drop in performance in the extrapolation setting, they simply treat it as one of the failure modes in NAQANet and provide no further analysis on this alarming issue on model reliability. A survey on numerical representations (Thawani et al., 2021) also mentions the extrapolation issue frequently found in these models, only to stop at reiterating the already identified issues.

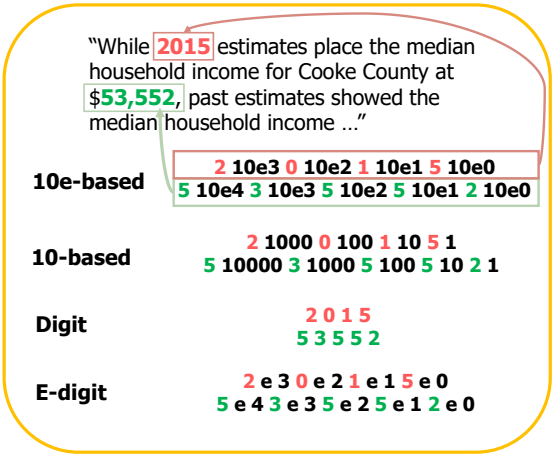

Figure 2: Illustration of how each surface form is represented and replaced in text.

A recent study by Nogueira et al. (2021) on the changes in how a number is presented to a model shows that different surface forms have significant influence on T5 (Raffel et al., 2020) in solving a simple arithmetic task. However, their proposed surface forms fail to extrapolate. They also explicitly provide the arithmetic operators and do not require complicated textual understanding for discrete reasoning. This begs the question of whether the same approach is viable in DROP, where reasoning is done across multiple sentences, requires dealing with heterogeneous type numbers, and the operations should be inferred from the text.

3 Empirical Investigation on Extrapolation

We first seek to determine whether the state-of-the-art models on DROP (Dua et al., 2019) can extrapolate their numerical reasoning capabilities to unseen numbers during inference.

Dataset DROP is a reading comprehension benchmark that requires models to perform a set of discrete reasoning such as counting, sorting and basic arithmetic operations. In this work, we construct the extrapolated version of DROP evaluation set by perturbing numbers with addition and multiplication of pre-defined numbers as in Figure 1. Then, we test the existing models for their extrapolation capabilities with these variant datasets.

Data Perturbation Prior to constructing the evaluation datasets, we use a named entity recog-

Model	Interpolate		Add(10)		Add(100)		Factor(10)		Factor(100)	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
NAQANet	46.20	49.24	40.87	43.61	38.31	40.34	30.17	34.09	23.03	26.01
NumNet	64.92	68.31	57.01	61.12	56.70	60.75	39.76	43.84	27.55	31.68
NumNet+(RoBERTa)	81.07	84.42	63.38	66.19	62.30	66.05	55.44	66.01	55.04	58.80
GenBERT	68.80	72.30	60.67	63.80	56.73	59.72	45.41	47.58	42.78	45.10

Table 1: The baseline and state-of-the-art DROP models tested on our four extrapolated versions of DROP evaluation dataset. **Add(N)** means adding N and **Factor(N)** means multiplying N to the numbers that appear in the passage. **Interpolate** shows the results on the original evaluation dataset.

Type	Count	Max	Min	Median	MAD
CARDINAL	33,732	48,466,928	0	31	140
MONEY	1,561	653,422,000	0.8	289.5	318.5
QUANTITY	23,072	4,000,160	1	27	19
DATE	30,549	105,000	0	1,907	191.5
TIME	2,627	160	0	13.5	9
PERCENT	5,123	280	0.01	12	14

Table 2: Type-wise analysis on numbers in DROP training dataset. MAD is the median absolute deviation.

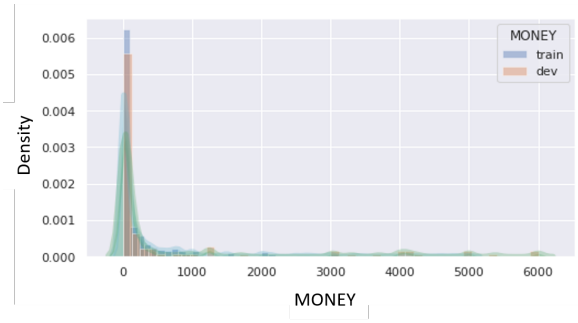

Figure 3: MONEY type number distribution in the DROP train and evaluation datasets. Bin width is set to 50 with the numbers shown up to 80th percentile for visibility. The numbers are highly skewed to right.

nition (NER) system[1] to extract and identify seven different entity types among the numbers in the text, namely: ORDINAL, DATE, QUANTITY, CARDINAL, MONEY, TIME, PERCENT. Among these seven entity types, we apply the aforementioned *extrapolation* perturbation to QUANTITY, CARDINAL and MONEY only, because DATE, PERCENT, TIME and ORDINAL require type-specific, handcrafted perturbations. For instance, if we were to perturb, "*King James was born in May 25, 1926*", it is not possible to simply change the range of "25" and "1926" by multiplying a 100, which neglects the entity-specific characteristics and requires question-level adjustment. Since we are probing the models to evaluate their extrapolation capability on unseen numbers, changing the range of the three types suffices. We use the four versions of extrapolated DROP evaluation set to observe the changes in performance along with the magnitude of changes in number range: Add(10), Add(100), Factor(10), Factor(100). Add(N) means adding N and Factor(N) means multiplying N to the numbers that appear in the passage.

The numbers from the passage, question and answer are perturbed with one of the four perturbation schemes above. Naturally, by the *distributive law*, the validity of the perturbed answer value holds. For example (see Figure 1), applying Factor(100) to a sequence, 49,927 + 18,009 + 12,182 = 80,128, results in 100 * (49,927 + 18,009 + 12,182) = 100 * (80,128). The same rule applies to other perturbation methods. As for the count-type answers that consist of numbers, we apply a heuristic where we consider both the number answers within the range of 0 to 9 and their extrapolated variants as answers. This prevents accidental perturbation of count-type answers and also considers arithmetic-type answers that have been extrapolated.

Models To inspect the extrapolation capability among the existing models in DROP, we evaluate the following representative models in the leaderboard: NAQANet (the official baseline model in DROP), NumNet (Ran et al., 2019), NumNet+(RoBERTa) and GenBERT (Geva et al., 2020). Although we mention QDGAT (Chen et al., 2020) in this paper, we did not evaluate it because its official implementation could not be reproduced.

Probing Models for Extrapolation We experiment on the models with the extrapolated DROP dataset and show that model performances degrade significantly as in Table 1. One notable observation from this experiment is that as the range of numbers increase ("*Factor(10)*" -> "*Factor(100)*"),

[1]Stanford's Stanza toolkit for NLP (Qi et al., 2020)

Surface Form	Number		Date		Others		All	
	EM	F1	EM	F1	EM	F1	EM	F1
Add(100)								
Original	62.05	62.06	33.12	34.44	48.97	57.02	56.73	59.72
10e-based	62.27	70.14	32.60	33.91	49.32	61.36	57.29	64.98
10-based	**62.55**	74.01	37.97	48.13	51.61	56.91	59.19	67.85
Digit	62.47	64.58	33.71	33.92	49.02	56.98	59.38	67.90
E-digit (Extrapolate)	62.42	**75.03**	**50.96**	**58.78**	**64.47**	**77.94**	**59.97**	**68.24**
Factor(100)								
Original	39.11	43.25	17.85	30.42	46.51	57.18	42.78	45.10
10e-based	40.31	45.06	18.59	35.79	49.59	60.74	43.02	49.94
10-based	40.76	45.90	32.48	38.23	50.27	62.11	49.24	56.47
Digit	41.64	45.82	17.97	32.38	51.74	62.72	44.97	51.76
E-digit (Extrapolate)	**61.56**	**63.72**	**41.40**	**55.74**	**55.48**	**64.68**	**57.91**	**63.98**

Table 3: Surface form evaluation with GenBERT on our **Add(100)** and **Factor(100)** extrapolated versions of DROP evaluation dataset. Our **E-digit (Extrapolate)** method outperforms all the other surface forms by a large margin.

model performances decrease accordingly. The result shows that even a small shift in the number range affects the model performance, implying that it is partly due to sample inefficiency. Meaning, that the perturbations create numbers that exist outside the number distribution in the training dataset, with the model trying to handle the vast coverage of numbers with typical subword representations. This problem is evident in the highly skewed number distribution in both the DROP training and evaluation dataset. The right-skewed distribution for the MONEY-type numbers in Figure 3, for example, shows a long tail, with the frequency of numbers in the training text quickly degrading as the magnitude of numbers grow (similar distribution is exhibited in CARDINAL, QUANTITY and PERCENT). This is also apparent in Table 2, where we see numbers that range up to millions but the median absolute deviation (MAD) is overly large for CARDINAL, MONEY and DATE. For TIME, PERCENT and QUANTITY, although we see a negligible spread, MAD's characteristic of ignoring the outliers like the MAX value of QUANTITY may have ignored less frequent, larger values. Such number distribution inhibits the models from generating an inductive bias for numbers, as the model is going to encounter only the numbers within the limited range during training. This lack of inductive bias for numbers prevents the model from extrapolating to out-of-distribution numbers in text. Thus, it is essential that the model gains a strong inductive bias for numbers, despite seeing numbers of arbitrary lengths.

4 Injecting Inductive Bias on Numbers with Surface Form Representations

After revealing the lack of extrapolation capability in the models, we gauge the influence of different surface forms of numbers as input to the MRC models. Based on the observation on the importance of surface forms in arithmetic word problems (AWP) (Nogueira et al., 2021), we evaluate if altering the surface form representation of numbers in DROP alleviates the performance discrepancy shown in Table 1. Moreover, we propose the new **E-digit** surface form to overcome the limitations of previous surface forms in extrapolation.

Surface Form Methods Our **E-digit** method makes use of two types of tokens, "*e*" and "*digit*", to reconstruct the numbers in the passage as in Figure 2. To elaborate, the **E-digit** method augments the typical digit-level number surface form by providing digit-position information with the *e* token and its corresponding *digit number* (See Figure 2). The three other surface forms proposed in Nogueira et al. (2021), namely 10e-based, 10-based and digit forms are composed of "*10e#*", "10^n" and numbers separated into digit-level representation, respectively. The principal difference between our **E-digit** and the three surface forms is that the *e* token embedding is digit-position independent, meaning it can occupy any digit-position as long as its followed by the digit-position number. On the contrary, 10e-based and 10-based methods

Model	EM	F1
GenBERT	68.80	72.30
E-digit (Interpolate)	68.14	71.05

Table 4: Comparison between the GenBERT model and its E-digit variant (i.e., E-digit(Interpolate)), which is trained with E-digit method and evaluated on the E-digit DROP dev set.

require a separate embedding for every digit position, with its number growing proportionally to the length of a number.

Here, we hypothesize that providing a position-independent token as in **E-digit** enables the model to leverage the "*e*" embedding to improve the extrapolation capability. We provide four versions of the original training dataset for the above four surface forms, and apply the same perturbation "*Factor(100)*" on evaluation set for inference. We use GenBERT as our proxy model because we need the model to generate answer texts like "2 e 2 7 e 1 0 e 0," whereas other models are incapable of generating calculated number answers in different surface forms, only performing span extraction and using special heads to assign {+, -, 0} on numbers appearing in the passage.

To validate the utility of the E-digit approach in the default, non-extrapolated setting, we compare the performance of the original GenBERT model against the E-digit(Interpolate) (Table 4), which is GenBERT fine-tuned with the E-digit method and evaluated on the original DROP evaluation set. Despite minor degradation in performance, the E-digit (Interpolate) performs comparably to GenBERT, which proves its effectiveness in representing numbers like digit tokenization does in the original GenBERT model. Our interpretation to such an outcome is that the performance gap is most likely caused by GenBERT's pre-training scheme (Geva et al., 2020), which employs digit subword inputs (*14* $\rightarrow$ *1 ##4*) to solve simple arithmetic problems to induce numerical reasoning skills. This may have caused the input mismatch issue since digit-level information explicitly provided by **E-digit** is absent during pre-training.

Analysis of Different Surface Forms The notable observation in Table 3 is that our **E-digit** method outperforms all the other surface forms, including the original model on the extrapolate DROP dataset. Also, the surface form methods all outperform the original models' subword tokenization approach. The results empirically show that: *(i)* providing digit information ("*e*", "*10e#*") along with numbers in their digit form is important in modeling numbers for extrapolation in a complicated textual reasoning task, and *(ii)* from the EM and F1 scores, we realize that the models still underperform in the extrapolation task when compared to the original interpolation task. The latter suggests that, in addition to the surface form problem identified in our work, there still are problems with the current approaches to number modeling in numerical MRC models.

Further analysis on the different answer types in DROP provides insight into the relationship between the answer types and surface forms. The **E-digit** method outperforms other forms notably in **Number** and **Date** categories. This shows that the "*e*" embedding learns to effectively represent numbers within the model despite seeing out-of-distribution numbers. The 10-based surface form, to our surprise, outperforms other surface forms in the **Date** type answers. We speculate that such a result arises from the year-type numbers' characteristic of typically ranging between numbers of 1000 to 2000, which enables the model to learn the relevance of the embedding "1000" to numbers in a year-related context. Overall, the **E-digit** surface form provides an explicit digit-level information of a number, which in turn empowers the model to effectively preserve and represent number information for numerical reasoning over text.

5 Conclusion

In this work, we investigated the extrapolation problem in complex numerical reasoning over text. Our probing results shed light on the significant lack of DROP models' capabilities by simulating a more realistic and ultimately needed benchmark (i.e., extrapolation). One of the key findings is that treating numbers as words inevitably requires a vast coverage of numbers, leading to sample inefficiency. This motivated us to adopt a more generalizable surface form representation, proposing the E-digit method that successfully generalizes to unseen numbers. Empirical results highlight simple surface representations benefit the model with digit information for extrapolation, and our E-digit method effectively generalizes it further. Our work opens up a new research direction in numerical reasoning over text on how to reduce the discrepancy between the original and extrapolated settings.

Acknowledgements

This work was supported by Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2013-2-00131, Development of Knowledge Evolutionary WiseQA Platform Technology for Human Knowledge Augmented Services).

References

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving bert a calculator: Finding operations and arguments with reading comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5949–5954.

Kunlong Chen, Weidi Xu, Xingyi Cheng, Zou Xiaochuan, Yuyu Zhang, Le Song, Taifeng Wang, Yuan Qi, and Wei Chu. 2020. Question directed graph attention network for numerical reasoning over text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6759–6768.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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 *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)*, pages 2368–2378.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2019. Neural module networks for reasoning over text. In *International Conference on Learning Representations*.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. A multi-type multi-span network for reading comprehension that requires discrete reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1596–1606.

Segwang Kim, Hyoungwook Nam, Jin-Tea Kim, and K. Jung. 2021. Neural sequence-to-grid module for learning symbolic rules. *ArXiv*, abs/2101.04921.

Andreas Madsen and Alexander Rosenberg Johansen. 2020. Neural arithmetic units. In *International Conference on Learning Representations*.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. A discrete hard em approach for weakly supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2844–2857.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Li. 2021. Investigating the limitations of the transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. Numnet: Machine reading comprehension with numerical reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2474–2484.

Amrita Saha, Shafiq Joty, and Steven CH Hoi. 2021. Weakly supervised neuro-symbolic module networks for numerical reasoning. *arXiv preprint arXiv:2101.11802*.

Dhanasekar Sundararaman, Shijing Si, Vivek Subramanian, Guoyin Wang, Devamanyu Hazarika, and Lawrence Carin. 2020. Methods for numeracy-preserving word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4742–4753.

Avijit Thawani, Jay Pujara, Filip Ilievski, and Pedro Szekely. 2021. Representing numbers in NLP: a survey and a vision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Lan-*

guage Technologies, pages 644–656, Online. Association for Computational Linguistics.

Andrew Trask, Felix Hill, Scott E Reed, Jack Rae, Chris Dyer, and Phil Blunsom. 2018. Neural arithmetic logic units. *Advances in Neural Information Processing Systems*, 31:8035–8044.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do nlp models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5310–5318.