Arithmetic-Based Pretraining – Improving Numeracy of Pretrained Language Models

Anonymous ACL submission

Abstract

State-of-the-art pretrained language models tend to perform below their capabilities when applied out-of-the-box on tasks that require understanding and working with numbers (usually referred to as numeracy). Recent work suggests two main reasons for this: (1) popular tokenisation algorithms have limited expressiveness for numbers, and (2) common pretraining objectives do not target numeracy. Approaches that address these shortcomings usually require architectural changes or pretraining from scratch. In this paper, we propose a new extended pretraining approach called Arithmetic-Based Pretraining that jointly addresses both of them in one extended pretraining step without requiring architectural changes or pretraining from scratch. Arithmetic-Based Pretraining combines (1) contrastive learning to improve the representation of numbers, and (2) a novel extended pretraining objective called Inferable Number Prediction Task to improve working with numbers. We evaluate our approach on three different tasks that require improved numeracy including (a) reading comprehension in the DROP dataset, (b) inference-on-tables in the InfoTabs dataset, and (c) table-to-text generation in WikiBio and SciGen datasets. Our results on DROP and InfoTabs show that our approach improves the accuracy by 9.6 and 33.9 points on these datasets, respectively. Our human evaluation on SciGen and WikiBio shows that our approach improves the factual correctness of generated outputs.[1]

1 Introduction

Numbers are ubiquitous in natural language. Therefore, understanding and working with numbers (usually referred to as numeracy) is a critical capability for pretrained language models such as BART (Lewis et al., 2020) or T5 (Raffel et al., 2019), cornerstones of modern NLP, in order to utilize quantitative information for various NLP tasks. Recent works question whether these models meet this requirement out-of-the-box (Wallace et al., 2019; Zhang et al., 2020): Common pretraining objectives such as the denoising autoencoder of BART (Lewis et al., 2020), the masked language modeling of BERT (Devlin et al., 2019), or span-corruption objective of T5 (Raffel et al., 2019), are designed for understanding structure and semantic meaning of language and not to learn working with numbers. Commonly used tokenisation algorithms such as Byte Pair Encoding (Sennrich et al., 2016) or WordPiece (Wu et al., 2016) are designed to handle patterns that are frequently observed during training, which is disadvantageous for numbers. For instance, 0.72 and 0.73 are two similar numbers. They should be processed similarly, but according to their frequency in the pretraining data they might be tokenised very differently, e.g., [0, ., 72] and [0, ., 7, 3], which will have an impact on their representation in embedding space.

Various approaches have been proposed recently to address these shortcomings, e.g., the character-level tokenisation to address the representation of numbers (Geva et al., 2020). However, most of them introduce additional components or rely on predefined features that limit their application, e.g., the proposed approach is only applicable for a specific task like reading comprehension (Andor et al., 2019) or table-to-text generation (Suadaa et al., 2021).

In this paper, we propose a new extended pretraining approach called Arithmetic-Based Pretraining that targets both shortcomings in one extended pretraining step using a combined loss function and without introducing new components or requiring pretraining from scratch. It consists of the following:

- A contrastive loss that uses both the character-level and subword-level tokenisation to improve the representation of numbers.

[1] Code and data are published here: `placeholder-url`.

- A denoising pretraining objective, called the Inferable Number Prediction Task, to improve the model's capability of working with numbers.

Our experiments show that arithmetic-based pretraining has a positive impact on BART (Lewis et al., 2020) and T5 (Raffel et al., 2019) in various tasks. It improves the accuracy in case of reading comprehension and inference-on-tables, and the factual correctness in case of table-to-text generation.

2 Related Work

Number Representations in Language Models. State-of-the-art language models like BART (Lewis et al., 2020) or T5 (Raffel et al., 2019) use subword-based tokenisation algorithms (such as Byte Pair Encoding (Sennrich et al., 2016)) to build vocabularies based on frequently observed sequences in a text corpus. While this is effective for common words, it is problematic for numbers. In an extensive study, Wallace et al. (2019) find that models using character-level tokenisation, such as ELMo (Peters et al., 2018), usually achieve better results in numerical probing tasks and extrapolate better to unseen numbers compared to models using subword-based tokenisation. Thawani et al. (2021), Peng et al. (2021) and Zhang et al. (2020) report similar findings. Therefore we use the character-level tokenisation for numbers to address this shortcoming in BART and T5.

Approaches for Improving Numeracy. Numeracy requires to understand and work with numbers, i.e., to do artihmetic operations, in order to generate the expected result. To improve this capability, recent approaches propose pretraining from scratch to tailor them towards specific tasks or to adjust the architecture of pretrained language models. TAPAS (Herzig et al., 2020) targets question answering with tabular data. It is pretrained from scratch and extends BERT (Devlin et al., 2019) by introducing additional embeddings for capturing tabular structure. GenBERT (Geva et al., 2020) reuses a pretrained BERT model and adds a decoder on top. It is then further trained using math word problems and arithmetic operations for (1) incorporating the character-level tokenisation for numbers into the model, and (2) to improve its numerical reasoning skills. It achieves state-of-the-art results on the DROP (Dua et al., 2019) and SQUAD (Rajpurkar et al., 2016) datasets. Andor et al. (2019) also reuses a pretrained BERT model, adds a new layer on top of BERT that predicts and executes arithmetic operations and targets reading comprehension. Suadaa et al. (2021) targets table-to-text generation and propose a framework that uses the template-guided text generation from Kale and Rastogi (2020) to inject pre-executed numerical operations into the pretrained GPT-2 (Radford et al., 2019) and T5 (Raffel et al., 2019) models. All these approaches result in new or task-specific models. With Arithmetic-Based Pretraining, we propose an approach that (1) is more cost effective than pretraining from scratch, as it can be used with already pretrained models, and (2) improves numeracy while maintaining the model's original purpose.

Domain-Adaptive Pretraining. The idea of domain-adaptive pretraining is to bridge the gap between the vocabulary of a model's original pretraining corpus and the target domain by continuing pretraining using in-domain data (Gururangan et al., 2020). In this work, we propose the Inferable Number Prediction Task which is similar to domain-adaptive pretraining if the data used is from the same domain as that of finetuning. However, we show that this is not the only reason for performance improvements (Section 5.3).

Contrastive Learning. Contrastive learning is a general way to learn to map vector representations of similar data points (usually called *anchor* and *positive*) close to each other while pushing non-similar data points apart. In NLP, it is commonly used for learning sentence representations (Kim et al., 2021; Giorgi et al., 2021) or semantic similarities (Wang et al., 2021). In this work, we use contrastive learning to improve the representation of numbers.

3 Arithmetic-Based Pretraining

In this section, we propose arithmetic-based pretraining to improve the numeracy of pretrained language models. It combines different tokenisation algorithms, i.e., character-level and subword-based, with contrastive learning to improve the representation of numbers in pretrained language models (Section 3.1), while training on the Inferable Number Prediction Task (Section 3.2) to improve the capability of working with numbers. Section 3.3 describes the joint loss function.

3.1 Contrastive Learning

We propose to use a contrastive loss as additional training signal to improve the representation of numbers. For example, the model should learn a similar representation for the number 108.89, whether it is initially tokenised as [1, 0, 8, ., 8, 9] (character-level) or [10, 8, ., 89] (subword-based). If a number is frequently occurred in the pretraining corpus, its corresponding subword-based encoding may be more informative. On the other hand, the character-level tokenisation is more informative for less frequent numbers. Therefore, our motivation is to benefit from both embedding spaces for learning better number representations. For implementation, we use the Multiple Negative Ranking Loss as proposed by Henderson et al. (2017)[2]:

$$\mathcal{L}_C = -\frac{1}{N}\sum_{i=1}^{N}\frac{e^{sim(avg(\hat{p}_i),avg(\hat{p\prime}_i))}}{\sum_j e^{sim(avg(\hat{p}_i),avg(\hat{p}_{neg}))}} \quad (1)$$

For the contrastive loss, we consider all numbers in the batch independently of the input sequences. Each number is used twice, once in character-level tokenisation (anchor), and once in subword-based tokenisation[3]. Assume p is a list of all numbers in the batch in character-level tokenisation. $p\prime$ is a list of all numbers in the batch in subword-based tokenisation. We consider p_i and $p\prime_i$ as a positive pair. Every other number in p and $p\prime$ is considered as negative sample to p_i (denoted as p_{neg}). $\hat{p}_i$, $\hat{p\prime}_i$, and $\hat{p}_{neg}$ are the corresponding embeddings after the encoder pass. sim represents the cosine similarity and avg represents the mean-average of the embedding. Averaging (avg) is a simple and effective form of aggregation which is necessary at this point, as the numbers are split into multiple tokens during tokenisation.

3.2 The Inferable Number Prediction Task

The Inferable Number Prediction Task is a variation of the classic masked language modeling objective (Devlin et al., 2019), but aims on improving a model's capability on working with numbers by focusing on data that requires arithmetic operations. The task consists of input C and the corresponding target sequence D. C consists of a pair of text sequences, C_1 and C_2, that are separated with a special character. C_2 equals to D, but contains a masked number that can be inferred from C_1. Given C, the task is to reconstruct D by correctly predicting the masked number in C_2[4]. For instance, for the task of table-to-text generation, C consists of the linearized form of the input table (C_1) and its description with one masked number (C_2). We select data with the following criteria:

- D and C_1 should have at least one overlapping entity, e.g., D should contain at least one of the entities that appear in the row or column headers of C_1 if C_1 is a table. This ensures that D is relevant to the information given in C_1.
- D should contain at least one number that either occurs in C_1 or is inferable by summation, subtraction, multiplication, division or ordering. This ensures that the masked number in C_2 is arithmetically related to the numbers given in C_1.

The selected data is then reduced to the necessary information. If C_1 is a table, we remove rows and columns that do not share entities with C_2. If C_1 is an extensive text or paragraph, we apply each of these heuristics to each of the sentences and retain only the matching ones (the same applies to C_2)[5].

For training, we use the cross-entropy loss function:

$$\mathcal{L}_{INP}(x, y) = \frac{1}{N}\sum_{n=1}^{N} -\log\left(\frac{e^{(x_{n,y_n})}}{\sum_{k=1}^{K} e^{(x_{n,k})}}\right) \quad (2)$$

where x represents the logits of the predicted input sequence, and $y = y_1, ..., y_N$ represents the indices of the tokens of the output sequence. N is the size of the target sequence. x_{n,y_n} is the logit of the x_n token corresponding to the output token y_n. K is the size of the model's vocabulary.

3.3 Joint Loss Function

We combine the contrastive loss $\mathcal{L}_C$ (Equation 1) and the loss for the Inferable Number Prediction Task $\mathcal{L}_{INP}$ (Equation 2) as weighted sum in a joint loss function:

$$\mathcal{L} = \frac{\mathcal{L}_C}{2} + \frac{\mathcal{L}_{INP}}{2} \quad (3)$$

[2]We use the implementation from the sentence-transformer library (Reimers and Gurevych, 2019).

[3]Note that we use both only for Arithmetic-Based Pretraining. For finetuning and during inference, we only use character-level tokenisation for numbers.

[4]Preliminary experiments revealed that just reconstructing the masked number, without its context, has a negative impact on a model's text generation capabilities.

[5]See Appendix B for further details and illustrations.

4 Experimental Setup

We implement our approach using Python 3.7, PyTorch (Paszke et al., 2019) and Huggingface (Wolf et al., 2020)). As pretrained language models, we use the large variant of BART (Lewis et al., 2020) and the base variant of T5 (Raffel et al., 2019) as provided by the Huggingface platform. T5 was pretrained in a multi-task scenario using task-specific prefixes. As we do not target such a scenario, we do not use these prefixes. We conduct all experiments on a Tesla V100-SXM3 GPU with 32 GB memory[6]. For experiments using table-to-text datasets, we represent tables as linearized sequence. We report the results of the best single runs.

4.1 Original Datasets

Reading Comprehension. The task of reading comprehension is to answer a question given a related text passage. The DROP dataset (Dua et al., 2019) is a reading comprehension dataset with over 96,000 questions in which answering questions requires discrete reasoning over text passages (e.g., arithmetic operations, orderings and comparisons). According to the authors, 59.1% of answers consist of numbers and therefore implicitly require performing arithmetic operations to be predicted correctly. Each paragraph consists of 9.19% numbers on average. We split the dev data into two equally-sized subsets and use one for testing. Each subset contains 4,828 questions.

Inference-on-Tables. Given a premise and a hypothesis, natural language inference (NLI) is the task of deciding whether the hypothesis is entailed, contradictory, or neutral to the premise. InfoTabs (Gupta et al., 2020) extends NLI to using semi-structured data, i.e., tables, as hypothesis. It is a crowdsourced dataset that consists of 23,738 hypothesis for 2,540 Wikipedia infoboxes from a variety of domains and provides three different test sets: one with data that is close to the distribution of the training data (in-domain), a cross-domain, and an adversarial test set. For the adversarial test set, the wording of hypotheses was slightly changed by expert annotators. Furthermore, they use another set of source tables for this test set, while retaining a distribution similar to the original training data. The cross-domain test set uses premises from domains that are not used for training, but generally require similar types of reasoning. According to the authors, InfoTabs requires the capability for a wide range of complex reasoning types across multiple rows, and that numerical and temporal reasoning (which implicitly requires performing arithmetic operations) is a significant part of this. Around 48% of dev data requires numerical or temporal reasoning. Each table consists of 13.89% numbers on average.

Table-to-Text Generation. Table-to-text generation is the task of summarizing tabular data in a descriptive text. As this data is often numerical, the challenging part of this task is to reason over numbers, i.e., to implicitly perform arithmetic operations such as ordering, summation or subtraction, or to capture magnitudes. SciGen (Moosavi et al., 2021) is a table-to-text generation dataset in which the input is a scientific table with its corresponding caption, and the output is the description of the table's content[7]. It is designed for arithmetic reasoning and consists of 53,136 annotated table-description pairs. Each table consists of 41.55% numbers on average.

WikiBio (Lebret et al., 2016) is a dataset from the biographical domain. The task is to reproduce the first paragraph of biographical Wikipedia articles, given the corresponding infobox. According to the authors, numbers (such as dates, ages and other quantities) play an important role. Each table consists of 16.83% numbers on average. However, most values can be directly copied from the tables and do not require mathematical operations. Overall, the dataset consists of 728,321 samples.

4.2 Preprocessing for the Inferable Number Prediction Task

To fulfill the requirements of the Inferable Number Prediction Task, we apply the criterias described in Section 3.2 to all datasets in an offline preprocessing step. In case of InfoTabs (Gupta et al., 2020), we only use the data labeled with entailed in order to exclude contradictions (see Appendix B for examples and illustrations).

Table 1 shows the resulting datasets. We also find that the resulting datasets have slightly different number to word ratios. In case of DROP (Dua et al., 2019) and InfoTabs, preprocessing increases the portion of numbers by 9.79% up to 18.98%,

[6] See Appendix A for details on hyperparameters.

[7] NumericNLG (Suadaa et al., 2021) is a similar dataset. As SciGen (Moosavi et al., 2021) provides more unsupervised training pairs that we can use for Arithmetic-Based Pretraining, we use SciGen in our experiments.

	Train	Dev	Test
SciGen	4,859	1,473	55
WikiBio	412,053	51,424	51,657
DROP	8,336	849	850
InfoTabs	1,981	1,800	1,800

Table 1: Data distribution for the Inferable Number Prediction Task after applying the criterias to the original dataset splits.

and 3.36% up to 17.25%, respectively. In case of WikiBio (Lebret et al., 2016) the ratio remains unchanged, but in case of SciGen (Moosavi et al., 2021), it reduces the portion of numbers per table by 7.67% to 33.88%.

	OCC	ORD	SUM	SUB	MUL	DIV
DROP	0.41	0.32	0.04	0.07	0.13	0.02
InfoTabs	0.23	0.34	0.05	0.17	0.15	0.06
SciGen	0.11	0.06	0.03	0.12	0.41	0.27
WikiBio	0.24	0.38	0.03	0.10	0.20	0.03

Table 2: Distribution of arithmetic operations across the datasets for this task.

Table 2 shows the ratio of samples per dataset that we have identified as being inferable by arithmetic operiations, i.e., occurence (OCC), ordering (ORD), summation (SUM), subtraction (SUB), multiplication (MUL) or division (DIV)[8].

5 Evaluation

In this section, we evaluate the impact of Arithmetic-Based Pretraining on downstream applications with BART (Lewis et al., 2020) and T5 (Raffel et al., 2019) using (1) in-domain data (Section 5.2), and (2) out-of-domain data (Section 5.3). For Arithmetic-Based Pretraining, we use the preprocessed subsets of the original datasets as described in Section 4.2.

5.1 Evaluation Metrics

For inference-on-tables, we evaluate the results using Exact Match (EM score). For reading comprehension, we additionally use F1 score. The EM score evaluates the prediction accuracy, i.e., if the prediction exactly matches the target, and is the preferred metric for these tasks (Dua et al., 2019; Gupta et al., 2020). The F1 score reports the overlap between the prediction and the target. This results in partial reward in cases where the prediction is partially correct. In case of table-to-text generation, we conduct a human evaluation. This is due to the shortcomings of common automatic metrics for this task, as they are hardly able to assess the correctness of information not directly contained in the source data, i.e., information obtained by reasoning (Moosavi et al., 2021; Chen et al., 2020a; Suadaa et al., 2021). [9]

5.2 In-Domain Pretraining

This section discusses the results on downstream tasks when using models that are pretrained using Arithmetic-Based Pretraining with in-domain data. *Baseline* represents the BART (Lewis et al., 2020) and T5 (Raffel et al., 2019) model directly finetuned on the corresponding dataset without Arithmetic-Based Pretraining. *Ours* represents these models with Arithmetic-Based Pretraining.

Reading Comprehension. Table 3 shows the results achieved on DROP (Dua et al., 2019). In case of BART (Lewis et al., 2020), Arithmetic-Based Pretraining increases the results by 9.6 points in EM score, indicating a large improvement in accuracy. Based on our analysis of the test results, i.e., by comparing the predictions of both models, we find that our approach reduces the incorrectly predicted numbers by 14.27% compared to the baseline. In case of T5 (Raffel et al., 2019), the results are in general much lower, but still our approach outperforms the baseline. Among other tasks, T5 was pretrained on reading comprehension using datasets similar to DROP (Raffel et al., 2019), e.g., MultiRC (Khashabi et al., 2018). We assume that not reusing the corresponding prefix may be a reason for lower performances of T5. However, based on our analysis of the test results, we find that our approach reduces incorrectly predicted numbers by 16.62%.

		EM	F1
DROP			
BART	Baseline	36.00	39.26
	Ours	**45.60**	**49.50**
T5	Baseline	10.40	14.60
	Ours	**11.00**	**15.20**

Table 3: Evaluation on the DROP dataset. Our approach outperforms the baseline in both cases.

Inference-on-Tables. Table 4 presents the prediction accuracies (EM score) achieved on the InfoTabs (Gupta et al., 2020) dataset.

For the in-domain test set, Arithmetic-Based Pretraining increases the EM score by 33.90 points in

[8] See Appendix C for a detailed analysis.

[9] Appendix D shows the results of the automatic metrics.

		In-Domain	Cross-Domain	Adversarial
BART	Baseline	33.30	23.67	27.68
	Ours	**67.20**	**54.40**	**57.20**
T5	Baseline	32.00	11.76	13.00
	Ours	**32.30**	**18.07**	**15.25**

Table 4: Evaluation on the InfoTabs dataset. This table shows the prediction accuracies (EM score) achieved on InfoTabs. Our approach outperforms the baseline in both cases.

case of BART (Lewis et al., 2020), which is a significant improvement in accuracy. Further analysis of the in-domain test results achieved with BART and our approach reveals that the model correctly predicts 60.30% of the entailments, 75.50% of the contradictions, and 65.83% of the neutrals. In case of T5 (Raffel et al., 2019), the improvements are rather negligible. This might again be due to not reusing the prefixes originally used for pretraining the model on similar tasks such as MNLI (Williams et al., 2017). Further analysis of the in-domain test results shows that T5 has a strong bias towards predicting entailment in both cases, i.e., the baseline and our approach. For the other two test sets, our approach also shows improvements over the baselines for BART and T5, indicating to improve the model's robustness and capability to extrapolate to unseen data.

Table-to-Text Generation. For human evaluation[10], we follow the approach used by Moosavi et al. (2021) for evaluating the results on SciGen. As this is very time-consuming, we only analyse 100 random table-description pairs from each, the SciGen and WikiBio (Lebret et al., 2016) dataset, and also only from the BART (Lewis et al., 2020) experiments. For SciGen, we use the results from the large split experiment.

For annotation, we break down each generated output to its corresponding statements (facts). We create one CSV file for each dataset that contains these statements in random order. This way, the annotator can not see whether a statement was generated by *Ours* (BART with Arithmetic-Based Pretraining) or by *Baseline* (BART without Arithmetic-Based Pretraining). Alongside with the generated statements, this CSV file contains the original tables and gold descriptions. Using this CSV file, the annotator then decides for each of the statements whether it belongs to one of the following labels:

[10]The human evaluation was conducted by one of the authors.

- *Entailed*: The statement is entailed in the gold description, e.g., a fact that is mentioned either in a similar or different wording in the description.
- *Extra*: The statement is not entailed in the gold description but is factually correct based on the table's content.
- *Incorrect*: The statement is relevant to the table, i.e., it contains relevant entities but is factually incorrect. For instance, the statement says *system A outperforms system B by 2 points* while based on the table system A has a lower performance than system B.
- *Hallucinated*: The statement is not relevant to the table.

Based on these labels, we then compute the recall (#entailed/#gold), precision (#entailed/#generated), correctness ((#entailed + #extra)/#generated), and hallucination (#hallucinated/#generated) scores for the generated facts. $\#gold$ and $\#generated$ refers to the respective number of included statements, not complete sequences. Table 5 shows the results.

	Prec.	Rec.	Cor.	Hall.
SciGen				
Baseline	0.08	0.02	0.31	**0.29**
Ours	**0.09**	**0.03**	**0.40**	0.33
WikiBio				
Baseline	0.22	0.07	0.33	0.03
Ours	**0.28**	**0.09**	**0.46**	**0.02**

Table 5: Results of the human evaluation. In both cases, our approach improves the correctness of the generated facts.

Arithmetic-Based Pretraining improves the precision, recall, and correctness for both SciGen and WikiBio. For WikiBio, it improves the precision by 0.06 points, suggesting that generated statements are more concise and closer to the target description. It also improves the ratio of statements that are factually correct by 0.13 points. This is similar for SciGen, although we observe a slight increase in hallucinations with our approach, which is a slight deterioration. We found that while Baseline seems to generate descriptions close to the target, Ours is somewhat more oriented towards the tabular values, whereby these values are used out-of-context in some cases which might be the reason for this

deterioration. Nevertheless, all models generate fluent and valid-looking descriptions (see Appendix G for examples).

5.3 Out-of-Domain Pretraining

To investigate whether the effectiveness of Arithmetic-based Pretraining is a result of using in-domain data for pretraining or improved numeracy, we evaluate our approach using out-of-domain data for pretraining. We focus on BART (Lewis et al., 2020) for this experiment and perform Arithmetic-Based Pretraining on a different dataset before finetuning on DROP (Dua et al., 2019) and InfoTabs (Gupta et al., 2020). For instance, for the DROP experiments, we pretrain models on WikiBio (Lebret et al., 2016), SciGen (Moosavi et al., 2021), and InfoTabs, which all include data from a different domain, before finetuning. For SciGen, we use the large split in this experiment. Table 6 shows the results.

	EM	F1
DROP		
Wikibio → DROP	6.00	33.50
InfoTabs → DROP	35.50	39.63
SciGen → DROP	**47.70**	51.60
DROP (in-domain)	45.60	49.50
InfoTabs		
WikiBio → InfoTabs	33.15	-
DROP → InfoTabs	32.80	-
SciGen → InfoTabs	64.70	-
InfoTabs (in-domain)	**67.20**	-

Table 6: Results of the out-of-domain pretraining. See Tables 3 and 4 for more details on the in-domain experiments.

Overall, the models pretrained using SciGen achieve the best out-of-domain results in both cases. In case of DROP, the results even exceed the ones achieved with in-domain pretraining. We suspect that the extent to which the pretraining dataset requires understanding and working with numbers has a major impact on the downstream performance. Among the datasets used, SciGen is in particular designed for the task of text generation based on arithmetic reasoning. It has by far the highest number to word ratio and the subset used for pretraining on the Inferable Number Prediction Task (see Section 4.2) predominantly depends on arithmetic operations such as multiplications or divisions (see Table 2) instead of lookups.

6 Ablation Study

In this section, we investigate the impact of using the character-level tokenisation for numbers, the contrastive loss and the Inferable Number Prediction Task (Section 3.2) on the overall effectiveness of Arithmetic-Based Pretraining. To focus on numeracy, the following experiments evaluate the number of correctly predicted masked numbers in the Inferable Number Prediction Task[11]. We use the preprocessed subsets of the original datasets for the Inferable Number Prediction Task (see Section 4.2). For evaluation, we use Exact Match (EM score) and F1 score (see Section 5.1). We start by investigating the impact of our masking procedure (masking only arithmetically related numbers), and continue with evaluating the impact of using the character-level tokenisation for numbers. The third experiment targets the effectiveness of our overall approach, Arithmetic-Based Pretraining, by additionally using the contrastive loss. The last experiment investigates the contribution of our masking procedure to the effectiveness of Arithmetic-Based Pretraining. Table 7 shows the results.

	EM	F1
WikiBio		
BART + DT + DM	29.69	48.12
CLT + INP	43.13	69.97
Ours	**77.38**	**74.69**
SciGen		
BART + DT + DM	7.04	32.21
DT + INP	7.20	35.11
CLT + INP	12.26	36.78
Ours	**24.68**	**45.81**
CLT + CL + DM	21.49	40.51
InfoTabs		
BART + DT + DM	12.43	22.17
DT + INP	23.20	46.17
CLT + INP	59.09	73.88
Ours	**60.45**	**74.33**
CLT + CL + DM	59.66	72.71
DROP		
BART + DT + DM	7.20	7.20
DT + INP	6.33	55.51
CLT + INP	29.40	66.43
Ours	**30.58**	**67.07**
CLT + CL + DM	25.37	59.83

Table 7: Ablation study on the Inferable Number Prediction Task. The combination of all constituents significantly outperforms the baseline in all experiments. We conduct *DT + INP* and *CLT + CL + DM* once for each task as ablation for the effectiveness of our masking procedure, and with SciGen (Moosavi et al., 2021) as representative for table-to-text generation.

[11] In case of the contrastive loss, we also experiment with other number representations (see Appendix E).

We consider BART with its default tokenisation (DT) and masking procedure (DM) as baseline (*BART+DT+DM*) for this experiment. *DT + INP* then uses the default tokenisation but our masking procedure (INP). In comparison with *BART+DT+DM*, the experiment shows that our masking procedure improves the results across all tasks. This is most significant in case of InfoTabs (up to 10.77 points in EM score). In case of DROP, it raises the F1 score from 7.20 to 55.51 points, meaning that there is a significantly larger overlap between predicted numbers and target numbers.

CLT + INP uses the character-level tokenisation (CLT) over the default tokenisation for numbers and again improves the results across all datasets, indicating improved capabilities for arithmetic operations. Compared to DT+INP, it improves the EM score by 35.89 points in case of InfoTabs, and by 23.07 points in case of DROP.

Ours finally combines CLT and INP with the contrastive loss (CL) as supporting signal to improve the representation of numbers and further improves the results across all datasets. This is most significant in case of the table-to-text datasets, where it improves the EM score by 34.25 points in case of WikiBio (Lebret et al., 2016), and 12.42 points in case of SciGen (Moosavi et al., 2021). Since we create the pairs for the contrastive loss batch-wise, i.e., we consider all numbers in a batch independently from the samples (see Section 3.1), an advantageous number to word ratio favors a good positive-negative pair ratio for the contrastive loss, as in the case of SciGen which has the highest number to word ratio in input tables (33.88%, see also Section 4.1). This is counteracted by WikiBio which has the lowest number to word ratio (16.32%). However, with 728,321 samples, Wikibio is the largest dataset. We therefore assume that more data compensates a poor number to word ratio.

CLT + CL + DM combines CLT with CL, but uses DM instead of INP and shows the contribution of our masking procedure to the effectiveness of Arithmetic-Based Pretraining. It deteriorates the EM score by 5.21 points in case of DROP, 3.19 points in case of SciGen, and 0.79 points in case of InfoTabs, showing that our masking procedure contributes a considerable share in the overall effectiveness of Arithmetic-Based Pretraining for improving the numeracy of a pretrained language model[12].

[12]We also did preliminary experiments with the math word problems dataset provided by Geva et al. (Geva et al., 2020) as a first pretraining task but found that this does not improve the results (see Appendix F).

7 Conclusions

In this paper, we propose Arithmetic-Based Pretraining, an approach for jointly addressing the shortcomings of pretrained language models in understanding and working with numbers (usually referred to as numeracy).

While existing approaches require architectural changes or pretraining from scratch, resulting in new or task-specific models, Arithmetic-Based Pretraining improves a model's numeracy while maintaining its original purpose. We use contrastive learning to benefit from both character-level and subword-based tokenisation to improve the representation of numbers, while training on a new pretraining objective, the Inferable Number Prediction Task, for improving the capabilities of performing arithmetic operations.

Our experiments and analysis indicate performance improvements due to better numeracy in different tasks and domains, including reading comprehension (DROP), inference-on-tables (InfoTabs), and table-to-text generation (e.g., WikiBio). In case of DROP, using our approach improves the results by 9.60 points in EM score over the BART baseline. For InfoTabs, our approach improves the results by 33.90 points, while also showing to be more robust against adversarial and out-of-domain evaluations. For table-to-text generation, our approach improves the correctness of generated facts over the BART baseline. Further experiments show that the effectiveness of our approach is not limited to in-domain pretraining, but also improves the results when pretrained with out-of-domain data. For example, pretraining on the SciGen dataset improves the results achieved on DROP when pretrained using in-domain data, i.e., the DROP dataset itself. Our ablation studies show that both contrastive learning and Inferable Number Prediction Task play an important role in the improved numeracy of the examined models.

8 Limitations

Our work is subject to some limitations. First of all, BART (Lewis et al., 2020) restricts the maximum length of input sequences to 1024 characters[13].

[13]https://huggingface.co/docs/transformers/model_doc/bart#transformers.BartConfig, last accessed on 10/10/22.

For better comparability, we also use T5 (Raffel et al., 2019) accordingly. This limitation is due to the increased computational complexity of longer input sequences, but it is problematic with table-to-text generation datasets. For example, SciGen (Moosavi et al., 2021) consists in large parts of tables that exceed this sequence length when represented as a linearized string. While we have tried to take this into account by reducing the input data to necessary information, it was not guaranteed that the model always sees the complete information, which certainly has a negative impact on the evaluation results achieved on the downstream tasks. We guess that the results would have been more expressive, if we would have used a different representation for tables, or focused on models that do not have this sequence length limitation.

Another limitation of our work concerns the impact of contrastive learning. According to Henderson et al. (2017), the impact of contrastive loss is favored by large batch sizes. Due to hardware limitations, we were only able to use small batch sizes (see Appendix A). The models might have adapted better if we would had the possibility to train with larger batch sizes. The next limitation is the way we use T5 in our experiments. The model was pretrained in a multi-task setup that also includes question answering, natural language inference and summarisation. In order to distinguish between these tasks, specific prefixes were used. As we do not address multi-task scenarios in this work, we did not reuse any of these prefixes for either Arithmetic-Based Pretraining or finetuning. We assume that this is the main reason for the large differences between the results with BART and T5 across many experiments. Maybe the model would have performed better, and more comparable to BART, if we would have used these prefixes.

Evaluation is also a critical point. Since it is not reliably traceable whether and which arithmetic operation was used by a model to come to a specific result, we can only infer improved capabilities for arithmetic operations by performance improvements in the Inferable Number Prediction Task. We cannot clearly distinguish performance improvements on specific arithmetic operations. Another limitation concerns the evaluation in table-to-text generation scenarios where generated descriptions usually suffer a high ratio of incorrect facts and hallucinations (Moosavi et al., 2021; Thawani et al., 2021; Chen et al., 2020b). This is not captured by automatic metrics. Although metrics such as PARENT (Dhingra et al., 2019) try to measure the factual correctness of generated descriptions, it requires a more individual examination in many cases. Especially in such highly specialized scenarios such as SciGen. Therefore, we conduct a human evaluation in order to analyse the impact of our Arithmetic-Based Pretraining on the downstream tasks. Due to limited resources, we were only able to conduct a small-scale human evaluation.

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 5947–5952, Hong Kong, China. Association for Computational Linguistics.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020a. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020b. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

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.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, Florence, Italy. 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, Minneapolis, Minnesota. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

John Giorgi, Osvald Nitski, Bo Wang, and Gary Bader. 2021. DeCLUTR: Deep contrastive learning for unsupervised textual representations. 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)*, pages 879–895, Online. Association for Computational Linguistics.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. INFOTABS: Inference on tables as semi-structured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324, Online. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Mihir Kale and Abhinav Rastogi. 2020. Template guided text generation for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6505–6520, Online. Association for Computational Linguistics.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. 2021. Self-guided contrastive learning for BERT sentence representations. 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)*, pages 2528–2540, Online. Association for Computational Linguistics.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Nafise Sadat Moosavi, Andreas Rücklé, Dan Roth, and Iryna Gurevych. 2021. Scigen: a dataset for reasoning-aware text generation from scientific tables. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, pages 8024–8035.

Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. 2021. Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. 2021. Towards table-to-text generation with numerical reasoning. 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)*, pages 1451–1465, Online. Association for Computational Linguistics.

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

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 5307–5315, Hong Kong, China. Association for Computational Linguistics.

Dong Wang, Ning Ding, Piji Li, and Haitao Zheng. 2021. CLINE: Contrastive learning with semantic negative examples for natural language understanding. 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)*, pages 2332–2342, Online. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xikun Zhang, Deepak Ramachandran, Ian Tenney, Yanai Elazar, and Dan Roth. 2020. Do language embeddings capture scales? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4889–4896, Online. Association for Computational Linguistics.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. 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 563–578, Hong Kong, China. Association for Computational Linguistics.

A Hyperparameters for Experiments

Table 8 shows the hyperparameter configuration for our experiments. In order to not train longer than necessary, we have determined the optimal number of epochs for each experiment by using early stopping with a patience of 10. For the downstream tasks, we have used the MoverScore (Zhao et al., 2019) with the table-to-text generation datasets. For DROP (Dua et al., 2019) and InfoTabs (Gupta et al., 2020), we have used the EM score. All models were trained for the same amount of epochs.

	Batch Size	Epochs	Learning Rate
Inferable Number Prediction Task			
SciGen	8	50	3e-5
WikiBio	8	3	3e-5
InfoTabs	8	21	3e-5
DROP	8	48	3e-5
Downstream Tasks			
SciGen	8	27	3e-5
WikiBio	8	9	3e-5
InfoTabs	8	14	3e-5
DROP	8	10	3e-5

Table 8: Hyperparameter Configuration.

B Inferable Number Prediction Task – Example Input Data

For table-to-text generation, Figure 1 shows an example of a (linearized) table from SciGen (Moosavi et al., 2021) with its caption (<*CAP*>) as C_1, concatenated with its masked description C_2 using </*s*>. <*s*> and </*s*> are special tokens used by BART (Lewis et al., 2020) to represent the beginning and ending of a sequence. In case of WikiBio (Lebret et al., 2016), the input data is represented accordingly.

<s> <R> <C> Model <C> F1 Score <C> Accuracy <R> <C> Our Approach <C> 76.58 <C> 88.55 <R> <C> Their Approach <C> 65.78 <C> 74.32 <CAP> Comparison between us and them. </s> Our approach achieves an F1 score **<mask>** points higher than their approach. </s>

Our approach achieves an F1 score **10.8** points higher than their approach.

Figure 1: Illustration of a linearized table that is used for the Inferable Number Prediction Task. <*R*>, <*C*> and <*CAP*> symbolize the beginning of a new row, cell, and the table's caption.

For DROP (Dua et al., 2019), Figure 2 shows an example. It consists of the paragraph C_1, and a question C_2. The question contains a number (2) that also occurs in the paragraph.

<s> He lied on the ground, motionless, for about 7 minutes before he was taken off the field on a cart. Dallas lead 12-10 with under 2 minutes to go. Dallas tried to come back, but Seattle forced a turnover on downs to end the game. </s> With less than **<mask>** minutes to go, how many points ahead was Dallas? </s>

With less than **2** minutes to go, how many points ahead was Dallas?

Figure 2: Illustration of an input sample for the Inferable Number Prediction Task using DROP.

Figure 3 shows an example for the InfoTabs (Gupta et al., 2020) datasets. It is basically the same as for the table-to-text generation datasets, but uses the hypothesis as C_2.

<s><R> <C> title <C> Country <C> Single match <C> Season <R> <C> India vs Pakistan 1999 <C> India <C> 465,000 (Five-day Test) India v. Pakistan at Eden Gardens, Kolkata, 16-20 February 1999 <C> 1,592,543 (Total), 26,528 per match, 2017 IPL </s> India faced Pakistan in a five day match in **<mask>**. </s>

India faced Pakistan in a five day match in **1999**.

Figure 3: Illustration of an input sample for the Inferable Number Prediction Task using InfoTabs.

C Inferable Number Prediction Task – Dataset Details

In this section, we want to provide more details on the distribution of arithmetic operations across datasets used for the Inferable Number Prediction Task. Table 9 shows the ratio of each arithmetic operation on the overall number of samples for each split for the InfoTabs (Gupta et al., 2020) dataset.

	OCC	ORD	SUM	SUB	MUL	DIV
Train	0.24	0.35	0.05	0.16	0.15	0.05
Dev	0.15	0.34	0.07	0.18	0.20	0.06
Test	0.22	0.16	0.09	0.23	0.23	0.07

Table 9: Ratio of arithmetic operations for each split of the InfoTabs dataset.

Table 10 shows this ratio for the DROP (Dua et al., 2019) dataset.

	OCC	ORD	SUM	SUB	MUL	DIV
Train	0.41	0.32	0.4	0.07	0.13	0.03
Dev	0.42	0.31	0.05	0.05	0.14	0.03
Test	0.43	0.30	0.04	0.05	0.15	0.03

Table 10: Ratio of arithmetic operations for each split of the DROP dataset.

Table 11 shows this ratio for the SciGen (Moosavi et al., 2021) dataset.

	OCC	ORD	SUM	SUB	MUL	DIV
Train	0.11	0.06	0.04	0.12	0.40	0.27
Dev	0.11	0.05	0.04	0.12	0.43	0.25
Test	0.15	0.09	0.02	0.19	0.43	0.13

Table 11: Ratio of arithmetic operations for each split of the SciGen dataset.

Table 12 shows this ratio for the WikiBio (Lebret et al., 2016) dataset.

	OCC	ORD	SUM	SUB	MUL	DIV
Train	0.25	0.38	0.03	0.10	0.20	0.03
Dev	0.25	0.38	0.03	0.10	0.19	0.04
Test	0.25	0.38	0.03	0.11	0.20	0.03

Table 12: Ratio of arithmetic operations for each split of the SciGen dataset.

D Evaluation Using Automatic Metrics

This section presents the evaluation of our results on table-to-text datasets using automatic metrics. For this, we use a variety of metrics commonly used for this task, i.e., *BLEU* (Papineni et al., 2002), *MoverScore* (Zhao et al., 2019), *BLEURT* (Sellam et al., 2020), and *PARENT* (Dhingra et al., 2019). While BLEU calculates the concordance between the predicted description and the actual target on word-level, MoverScore and BLEURT measure the semantic concordance between the predicted description and the target using BERT (Devlin et al., 2019). BLEURT also takes the fluency of the predictions into account. PARENT estimates the factual correctness by comparing the predicted description to the original table and the target description, and especially rewards correct information that is contained in the table but not in the target. It has a higher correlation with human judgement. Table 13 reports the results.

Based on PARENT and MoverScore, our approach is slightly superior in six out of ten experiments. However, based on these results, it is not really possible to conclude an improvement that can be directly attributed to Arithmetic-Based Pretraining in most cases, as none of these metrics can really assess the correctness of a fact that might be reasoned from the source data (Moosavi et al., 2021; Chen et al., 2020a; Suadaa et al., 2021). However, PARENT tries to address this, which is why this metric is the most appropriate. Like BLEURT, Moverscore measures the semantic concordance between target and prediction. The advantage of MoverScore is that it is easier to interpret.

			MoverS	BLEU	BLEURT	PARENT
SciGen						
BART	Baseline	Few	52.48	4.60	-0.63	3.38
		Medium	53.76	4.26	-0.69	3.72
		Large	53.43	4.87	-0.70	3.68
	Ours	Few	**53.30**	1.73	-0.76	**3.81**
		Medium	53.40	2.71	-0.78	3.45
		Large	**55.00**	9.30	-0.76	**3.82**
T5	Baseline	Few	52.30	2.96	-0.94	6.39
		Medium	51.79	2.67	-0.95	4.08
		Large	53.00	3.40	-0.70	5.18
	Ours	Few	52.00	2.83	-0.98	4.32
		Medium	**52.00**	2.51	-0.86	**4.70**
		Large	**53.40**	2.96	-0.89	**6.72**
WikiBio						
BART	Baseline		61.50	17.98	-0.64	45.18
	Ours		**62.78**	18.54	-0.27	44.32
T5	Baseline		60.30	17.94	-0.86	43.97
	Ours		60.10	20.00	-0.22	**45.25**

Table 13: Evaluation of our results on table-to-text datasets using automatic metrics. *Baseline* presents the result of the BART-large and T5-base models without Arithmetic-Based Pretraining. *Ours* show the result of these models with Arithmetic-Based Pretraining. Results of PARENT and MoverScore are highlighted. PARENT is the most appropriate metric for our approach.

E Experiments using other Contrastive Representations

Regarding the contrastive representation, we also experiment with number representations other than the default subword-level one in order to improve the representation of numbers using the character-level tokenisation, i.e., exponent-mantissa (Zhang et al., 2020), a verbalized representation, and a combination of all of them (across multiple batches) using the Inferable Number Prediction Task. We focus on BART (Lewis et al., 2020) for this experiment. We conduct this experiment using the large split of the SciGen dataset (Moosavi et al., 2021). Table 14 shows the results.

None of the other representations improves the results over using the default subword-level tokenisation.

Experiment	EM	F1
BART (verb. repr.)	15.69	41.01
BART (exp.-mant. repr)	18.13	36.78
BART (subword-based tok.)	**24.68**	**45.81**
BART (combined)	17.92	38.43

Table 14: Comparison of results when using different representations for incorporating the character-level tokenisation.

F Preliminary Math Experiments

With GenBERT, Geva et al. (2020) propose to start pretraining with math word problems in order to improve the model's number understanding and capabilities for arithmetic operations. Therefore, following this idea would be an obvious step in order to improve the numeracy of general purpose pretrained language models. Table 15 shows the results of a preliminary experiment using GenBERT's math word problems dataset (MWP), BART (Lewis et al., 2020) and SciGen (Moosavi et al., 2021) on the Inferable Number Prediction Task.

Experiment	EM	F1
Baseline	7.20	35.11
MWP-pretrained Baseline	15.19	34.18
MWP-pretrained Baseline + CLT	22.94	42.55
MWP-pretrained Baseline + CLT + CL	22.78	43.14
Ours	**24.68**	**45.81**

Table 15: Results achieved on the Inferable Number Prediction Task with and without pretraining using math word problems.

Baseline refers to the default BART model. *MWP-pretrained Baseline* shows the results for *Baseline*, but further pretrained on MWP. *MWP-pretrained Baseline + CLT* represents the results for *MWP-pretrained Baseline*, but uses the character-level representation (CLT) for numbers instead of BART's default tokenisation. Accordingly, *MWP-pretrained Baseline + CLT + CL* incorporates the contrastive loss (CL) as additional training signal. The results show that pretraining using math word problems as a first step in general improves the results for the Inferable Number Prediction Task, but not over using Arithmetic-Based Pretraining (*Ours*).

In case of SciGen, the Inferable Number Prediction Task, only uses samples with target descriptions that contain numbers that are inferable from the input table by lookup or arithmetic operations (see Section 4.2). Therefore, even though it is a synthetic task, the results give insights on how effective pretraining on math word problems is for improving a model's numeracy.

G Examples from the Human Evaluation

Figure 4 shows two sample generations from our approach and the BART (Lewis et al., 2020) baseline from the SciGen (Moosavi et al., 2021) experiment using the medium split. Both read fluent and plausible.

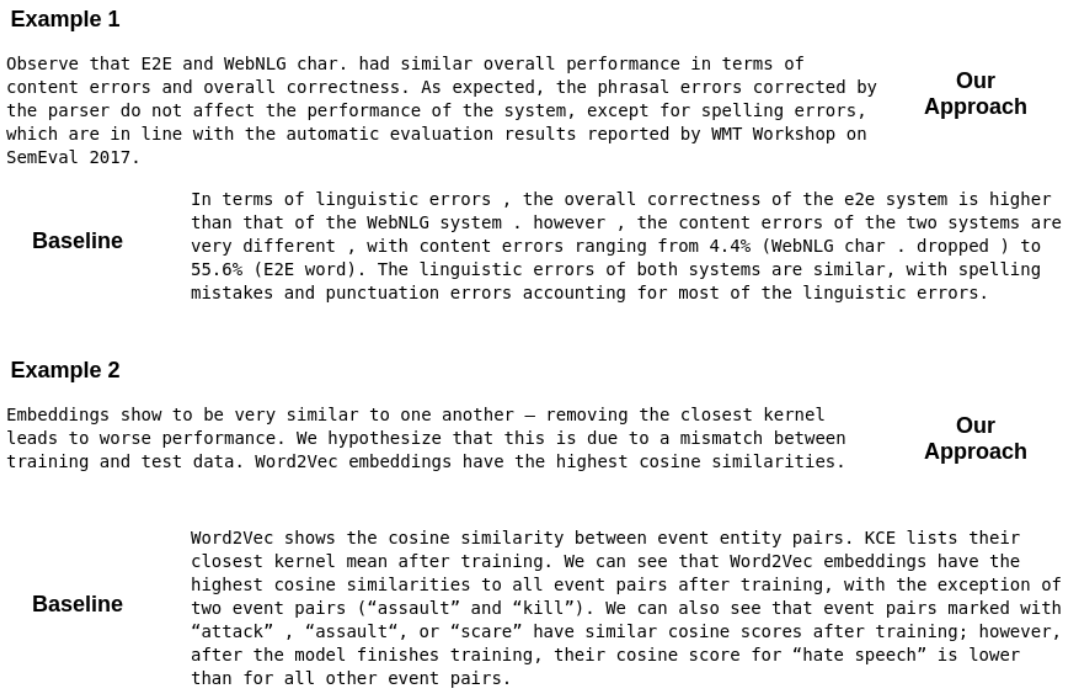

Figure 4: Generation from our approach and the BART-large baseline from the SciGen experiment using the medium split.