Arithmetic-Based Pretraining – Improving Numeracy of Pretrained Language Models

Dominic Petrak[†] , **Nafise Sadat Moosavi**[‡], **Iryna Gurevych**[†]

[†]Ubiquitous Knowledge Processing Lab (UKP Lab),
Department of Computer Science and Hessian Center for AI (hessian.AI),
Technical University of Darmstadt, Germany
https://www.ukp.tu-darmstadt.de
[‡]Department of Computer Science, The University of Sheffield, UK

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. 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 in one extended pretraining step without requiring architectural changes or pretraining from scratch. Arithmetic-Based Pretraining combines contrastive learning to improve the number representation, and a novel extended pretraining objective called Inferable Number Prediction Task to improve numeracy. Our experiments show the effectiveness of Arithmetic-Based Pretraining in three different tasks that require improved numeracy, i.e., reading comprehension in the DROP dataset, inference-on-tables in the InfoTabs dataset, and table-to-text generation in the WikiBio and SciGen datasets[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 objective of BERT (Devlin et al., 2019), or the 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. Furthermore, commonly used subword-based 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. To address these shortcomings, various approaches have been proposed recently. However, most of them introduce additional components or rely on predefined features that limit their application, e.g., they are only applicable in a specific task like reading comprehension (Andor et al., 2019; Geva et al., 2020) or require architectural changes (Herzig et al., 2020).

In this paper, we propose a new extended pretraining approach called Arithmetic-Based Pretraining that targets both shortcomings for pretrained language models in one extended pretraining step without introducing new components or requiring pretraining from scratch. It consists of:

- A contrastive loss that combines subword-based with character-level tokenisation to improve the representation of numbers.
- 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

[1]Code, data, and models trained using Arithmetic-Based Pretraining are available here: https://github.com/UKPLab/starsem2023-arithmetic-based-pretraining.

et al., 2020), T5 (Raffel et al., 2019) and Flan-T5 (Chung et al., 2022) 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) shows 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. In our work, we use the character-level tokenisation for numbers to address this shortcoming in BART, T5, and Flan-T5 (Chung et al., 2022).

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 or architectural changes to tailor pretrained language models towards specific tasks. 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, and (2) to improve the numerical reasoning skills. It achieves state-of-the-art results in the DROP (Dua et al., 2019) and SQUAD (Rajpurkar et al., 2016) datasets. Andor et al. (2019) also reuses the pretrained BERT model and targets reading comprehension. They add a new layer on top that predicts and executes arithmetic operations. Suadaa et al. (2021) target 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.

In their experiments, all of these works show that much of their performance improvements are due to specific design decisions or multi-level pretraining setups which result in new or task-specific models. With Arithmetic-Based Pretraining, we propose an approach that improves a model's numeracy with just one extended pretraining step and without changing its architecture.

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. 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 frequently occurs in the pretraining corpus, its corresponding subword-based encoding may be more informative. If this is not the case, its character-level tokenisation may be more informative. 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]:

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

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 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 (C_2 in C) 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 (C_2 in C) 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.

Next, we reduce C to the necessary information. 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). If C_1 is a table, we remove rows and columns that do not share entities with C_2 (see Appendix B for further details and illustrations).

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

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

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:

$$\mathcal{L} = \frac{\mathcal{L}_C}{2} + \frac{\mathcal{L}_{INP}}{2} \tag{3}$$

[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.

4 Experimental Setup

We implement our approach using Python 3.10, 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) and Flan-T5 (Chung et al., 2022) as provided by the Huggingface platform (see Appendix A for details on hyperparameters)[5]. All models are pretrained Transformer-based encoder-decoder models, but different in size. BART-large consists of a total of 24 layers and 406M parameters. T5-base and Flan-T5-base consist of 12 layers and 220M parameters. Flan-T5 is based on T5, but trained on more tasks, e.g., arithmetic reasoning, and chain-of-thought data (instructions). It significantly improves the results of the original model in many tasks (Chung et al., 2022). We conduct all experiments on a Tesla V100-SXM3 GPU with 32 GB memory. 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 by reasoning over a related text passage. DROP (Dua et al., 2019) is such a dataset. It contains over 96,567 open-domain question-answer pairs and 6,735 paragraphs. 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 question-answer pairs.

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 consists of 23,738 hypothesis for 2,540 Wikipedia infoboxes from a variety of domains and provides three different test sets: in-domain, cross-domain, and an adversarial test set. The cross-domain test set uses premises from domains not used for training. The adversarial test set uses a different set of source tables. Furthermore, the wording of hypotheses was slightly changed by expert annotators. According to the authors, InfoTabs requires numerical and temporal reasoning (which implicitly requires performing arithmetic operations) across multiple rows and to a large extent. Each table consists on average of $13,89\%$ numbers.

[5]We could not use the large variant of T5 and Flan-T5 due to hardware limitations (each model has 770M parameters).

Table-to-Text Generation. Table-to-text generation is the task of summarizing tabular data (which is often numerical) in a descriptive text. It requires 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 that requires to generate descriptions for scientific tables[6]. It is designed for arithmetic reasoning and consists of 53,136 table-description pairs. Each table consists of 41.55% numbers on average.

WikiBio (Lebret et al., 2016) is a dataset from the biographical domain. It consists of 728,321 table-description pairs. The task is to reproduce the first paragraph of biographical Wikipedia articles, given the corresponding infobox. According to the authors, 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 arithmetic operations.

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.

	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.

[6]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.

We also find that the resulting datasets have slightly different number-to-word ratios. In the case of DROP (Dua et al., 2019) and InfoTabs, preprocessing increases the portion of numbers up to 18.98% and 17.25% in paragraphs and tables. In the case of WikiBio (Lebret et al., 2016) the ratio remains unchanged and in the case of SciGen (Moosavi et al., 2021) it reduces the numbers per table 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 in the preprocessed datasets.

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). Appendix C provides a detailed analysis.

5 Evaluation

In this section, we evaluate the impact of Arithmetic-Based Pretraining on downstream applications with BART (Lewis et al., 2020), T5 (Raffel et al., 2019) and Flan-T5 (Chung et al., 2022) using in-domain data (Section 5.2), and 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. It 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., 2020b; Suadaa et al., 2021). We provide the results of the automatic metrics in Appendix D.

For all experiments, *Baseline* represents the BART (Lewis et al., 2020), T5 (Raffel et al., 2019), and Flan-T5 (Chung et al., 2022) model directly finetuned on the corresponding dataset without Arithmetic-Based Pretraining. *Ours* represents these models with Arithmetic-Based Pretraining. We highlight statistically significant improvements of Ours over the respective baseline in the tables (independent two-sample t-test, $p \leq 0.05$).

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. For comparison, we will also report the results of the specialised state-of-the-art model for each task.

Reading Comprehension. Table 3 shows the results achieved on DROP (Dua et al., 2019).

		EM	**F1**
BART	Baseline	36.00	39.26
	Ours	**45.60**	**49.50**
T5	Baseline	10.40	14.60
	Ours	11.00	15.20
Flan-T5	Baseline	46.34	94.41
	Ours	**72.18**	**97.65**
QDCAT		85.46	88.38

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

In all cases, Arithmetic-Based Pretraining improves the results over the baseline. Based on our analysis of the test results, i.e., by comparing the predictions of Baseline with Ours, we find that our approach reduces the incorrectly predicted numbers by 14.27% in case of BART (Lewis et al., 2020), 16.62% in case of T5 (Raffel et al., 2019), and 30.56% in case of Flan-T5 (Chung et al., 2022). The results achieved with Flan-T5 even outperform the results reported by Geva et al. (2020) for GenBERT (EM 68.6)[7]. Regarding the performance differences between BART and T5, we attribute this to the difference in model size. In this context, the performance difference between BART and Flan-T5 is particularly interesting. We attribute this to the fact that among other things, Flan-T5 was trained in arithmetic reasoning. QDCAT (Chen

[7]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 G).

et al., 2020a) is the current state-of-the-art in the DROP task. It was built for reading comprehension and is based on RoBERTa (Liu et al., 2019), but adds an additional question-conditioned reasoning step on top (using a graph-attention network).

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

		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**
Flan-T5	Baseline	27.23	25.14	29.17
	Ours	**34.04**	26.14	29.04
BPR		78.42	71.97	70.03

Table 4: Evaluation on the InfoTabs dataset. Our approach significantly improves the results on the in-domain data.

Similarly to reading comprehension, Arithmetic-Based Pretraining significantly improves EM scores in all cases. This applies especially to the in-domain test set. For the other two test sets, our approach also shows improvements over the baselines (mostly for BART (Lewis et al., 2020)), indicating to improve the model's robustness and capability to extrapolate to unseen data. We attribute performance differences to model sizes. Furthermore, analysis of the in-domain test results shows that T5 and Flan-T5 are biased toward predicting entailment. Since we observe this in both Baseline and Ours, we do not attribute this to how the data was preprocessed for the Inferable Number Prediction Task (Section 4.2). This is different for BART. An analysis of the in-domain test results shows that the model correctly predicts 60.30% of entailments, 75.50% of contradictions, and 65.83% of neutrals. BPR (Neeraja et al., 2021) is the current state-of-the-art in the InfoTabs task. It is based on BERT (Devlin et al., 2019) but built for inference over tabular data. It provides an improved representation of the input data, is pretrained on MultiNLI (Williams et al., 2018), and incorporates external knowledge.

Table-to-Text Generation. For human evaluation[8], 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[9].

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 Baseline (BART without Arithmetic-Based Pretraining). Alongside with the generated statements, this CSV file contains the original tables and gold descriptions. The annotator then decides for each of the statements whether it belongs to one of the following labels:

- *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.

Arithmetic-Based Pretraining improves the precision, recall, and correctness for both SciGen and WikiBio. In case of WikiBio, it improves the precision by 0.06 points, suggesting that generated

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

[9]For SciGen, BART is the current state-of-the-art, and the baseline results of our human evaluation are comparable with those reported by Moosavi et al. (2021). We are not aware of a comparable human evaluation for WikiBio. Appendix D shows a comparison of automatic metrics for both datasets.

	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.

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. In case of SciGen, the baseline results reflect the results reported by Moosavi et al. (2021), who also used the large variant of BART for their experiments. Ours improves the results in almost every aspect (especially in case of factual correctness, where it improves the results by 0.09 points). However, we observe a slight increase in hallucinations, which is a minor 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 H for examples). This suggests that Arithmetic-Based Pretraining has no negative impact on a model's text generation capability. This is also supported by the results achieved using automatic metrics (see Appendix D).

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 (domain-adaptive 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. Overall, the models pretrained using SciGen achieve the best out-of-

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

Table 6: Results of the out-of-domain pretraining (see Tables 3 and 4 for the in-domain experiments).

domain results in both cases. In case of DROP, the results even exceed the ones achieved with in-domain pretraining. We find that the extent to which the pretraining dataset requires understanding and working with numbers has a major impact on the downstream performance (the more, the greater the impact). Among the datasets used, SciGen is in particular designed for the task of text generation based on arithmetic reasoning. It has a high number-to-word ratio and the subset used for pretraining on the Inferable Number Prediction Task (see Section 3.2) predominantly depends on arithmetic operations such as multiplications or divisions (see Table 2) instead of lookups or orderings (like in the other datasets).

6 Ablation Study

In this section, we investigate the impact of Arithmetic-Based Pretraining on the numeracy of a pretrained language model. Due to the shortcomings of automatic metrics in table-to-text generation (see Section 5.1) and because we want to be able to compare and discuss the impact of each component across datasets, we use the Inferable Number Prediction task for this and evaluate the number of correctly predicted context-related masked numbers (please see Appendix E for ablation experiments in downstream tasks)[10]. 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). Table 7 shows the results.

We consider the large variant of BART (Lewis et al., 2020) with its default tokenisation (DT) and masking procedure (DM) as baseline for this

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

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

Table 7: Ablation study on the Inferable Number Prediction Task. We conduct DT + INP and Ours - INP once for each task and with SciGen (Moosavi et al., 2021) as representative for table-to-text generation.

experiment. *DT + INP* uses the default tokenisation but our masking procedure (INP). *CLT + INP* then uses the character-level tokenisation for numbers (CLT). *Ours* finally combines CLT and INP with the contrastive loss (CL) as supporting signal to improve the representation of numbers. As last ablation, *Ours - INP* combines CLT with the contrastive loss but uses DM instead of INP and shows the contribution of our masking procedure to the effectiveness of Arithmetic-Based Pretraining.

In comparison with BART, DT + INP 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. Using character-level instead of default tokenisation for numbers (CLT + INP) 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 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 a lower number-to-word ratio (16.32%). However, with 728,321 samples, Wikibio is the largest dataset. We therefore assume that more data compensates for a poor number-to-word ratio. Ours - INP 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. This shows the contribution of our masking procedure to the effectiveness of Arithmetic-Based Pretraining.

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). In contrast to existing approaches, Arithmetic-Based Pretraining does not require architectural changes or pretraining from scratch. It uses contrastive learning to improve number representation and a novel extended pretraining objective, the Inferable Number Prediction Task, to improve numeracy in just one extended pretraining step. Our experiments show performance improvements due to better numeracy in three different state-of-the-art pretrained language models, BART, T5, and Flan-T5, across various tasks and domains, including reading comprehension (DROP), inference-on-tables (InfoTabs), and table-to-text generation (SciGen and WikiBio). We show that the effectiveness of our approach is not limited to in-domain pretraining, but rather depends on the extent to which the dataset used in the Inferable Number Prediction Task requires understanding numbers. For example, pretraining on the SciGen dataset improves the results achieved on DROP. Our ablation studies show that contrastive learning and the Inferable Number Prediction Task are key to improving the numeracy of the examined models.

8 Limitations

Our work is subject to some limitations. First of all, due to hardware limitations, we could not use the large variant of T5 (Raffel et al., 2019) and Flan-T5 (Chung et al., 2022) in a setting comparable to our BART-large experiments. Furthermore, BART (Lewis et al., 2020) restricts the maximum length of input sequences to 1024 characters[11]. For better comparability, we also use T5 and Flan-T5 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 sequence. 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. Regarding the weighting of contrastive and masked loss in the joint loss function, we only use equal weighting for our experiments, since we found that this already leads to good results, and due to the already large number of experiments conducted in this paper, we did not experiment with other weightings. However, optimizing this hyperparameter could further improve the results.

Evaluation is also a critical point. 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. However, due to limited resources, we were only able to conduct a small-scale human evaluation. At this point, we would also like to mention that our evaluation setup in general is subject to limitations. As an extended pretraining approach, Arithmetic-Based Pretraining might have a negative impact on a model's general applicability, i.e., downstream performance in tasks used for pretraining, e.g., translation in case of T5, or other non-number related tasks commonly used in model benchmarking, such as question answering, text classification, or sentiment analysis. We only examined the impact on text generation as part of our human evaluation and with automatic metrics (see Appendix D). However, since (1) the Inferable Number Prediction Task (Section 3.2) is a variation of the widely used masked language modeling objective (Devlin et al., 2019), and (2) character-level tokenisation does not introduce new embeddings into a pretrained language model, we don't expect a negative impact here.

Another limitation concerns the evaluation of the Inferable Number Prediction Task on a model's numeracy. 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.

[11] `https://huggingface.co/docs/transformers/model_doc/bart#transformers.BartConfig`, last accessed on 10/02/23.

9 Acknowledgements

This research work has been funded by the German Federal Ministry of Education and Research and the Hessian Ministry of Higher Education, Research, Science and the Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE. It also received funding from the German Research Foundation (DFG) under grant № EC 503/1-1 and GU 798/21-1.

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.

Kunlong Chen, Weidi Xu, Xingyi Cheng, Zou Xiaochuan, Yuyu Zhang, Le Song, Taifeng Wang, Yuan Qi, and Wei Chu. 2020a. 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, 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.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

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. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

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.

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.

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

Nafise Moosavi, Andreas Rücklé, Dan Roth, and Iryna Gurevych. 2021. Scigen: a dataset for reasoning-aware text generation from scientific tables. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

J. Neeraja, Vivek Gupta, and Vivek Srikumar. 2021. Incorporating external knowledge to enhance tabular reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2799–2809, Online. Association for Computational Linguistics.

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, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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.

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.

Clément Rebuffel, Marco Roberti, Laure Soulier, Geoffrey Scoutheeten, Rossella Cancelliere, and Patrick Gallinari. 2021. Controlling hallucinations at word level in data-to-text generation.

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 Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

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

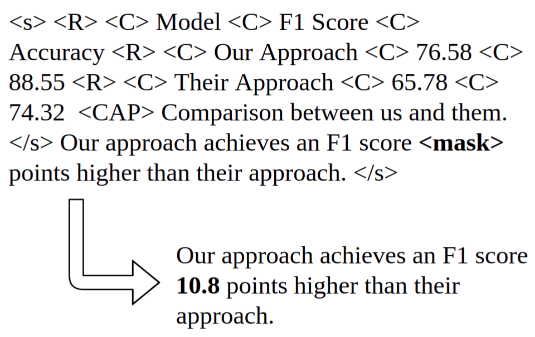
Table 8 shows the hyperparameter configuration for our experiments. In order to not train longer than necessary, we have determined the optimal number of epochs for each experiment by using early stopping with a patience of 10. For the downstream tasks, we have used the MoverScore (Zhao et al., 2019) 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 as C_1, concatenated to 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.

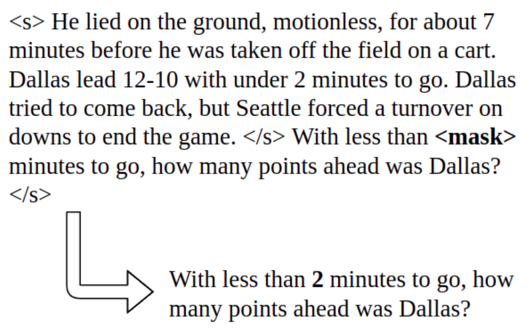


<s> <R> <C> Model <C> F1 Score <C> Accuracy <R> <C> Our Approach <C> 76.58 <C> 88.55 <R> <C> Their Approach <C> 65.78 <C> 74.32 <CAP> Comparison between us and them. </s> Our approach achieves an F1 score **<mask>** points higher than their approach. </s>

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.

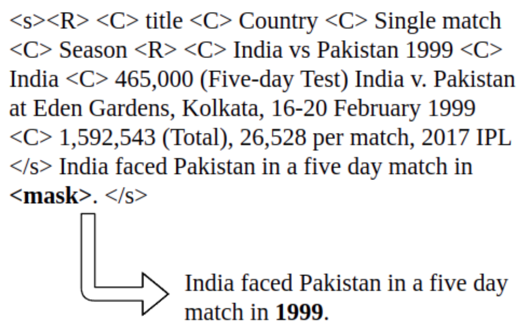
For DROP (Dua et al., 2019), Figure 2 shows an example. It consists of the paragraph C_1, and a question C_2. The question contains a number (2) that also occurs in the paragraph.


<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 judgment. Table 13 reports the results. We highlight statistically significant improvements of our approach over the respective baseline in the tables (independent two-sample t-test, $p \leq 0.05$).

			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
	BART (Moosavi et al.)	Large	14.00	5.04	-0.71	-
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**
	BART (Moosavi et el.)	Large	6.00	3.38	-0.79	-
Flan-T5	Baseline	Few	53.03	2.76	-0.67	7.89
		Medium	53.56	3.03	-0.68	6.14
		Large	54.15	3.54	-0.65	7.94
	Ours	Few	**54.22**	3.14	-0.65	8.54
		Medium	**54.76**	3.25	-0.71	**8.12**
		Large	**55.12**	3.34	-0.61	**9.32**
			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**
Flan-T5	Baseline		59.81	17.56	-0.78	44.67
	Ours		**62.51**	**21.11**	**-0.18**	**46.10**
	MBD		-	41.56	-	56.16

Table 13: Evaluation of our results on table-to-text datasets using automatic metrics. *Baseline* presents the results of the BART-large and Flan-T5-base models without Arithmetic-Based Pretraining. *Ours* shows the results of these models with Arithmetic-Based Pretraining.

The results show that Arithmetic-Based Pretraining slightly improves the performance in most experiments (based on PARENT and MoverScore), and has no negative impact text generation capabilities. However, as outlined in Section 5.1, 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., 2020b; Suadaa et al., 2021). PARENT tries to address this, which is why this metric is the most appropriate one. Like BLEURT, Moverscore measures the semantic concordance between target and prediction. The advantage of MoverScore is that it is easier to interpret.

In case of SciGen, even our baseline results for BART (Lewis et al., 2020) are better than reported by Moosavi et al. (2021). We attribute this to different training hyperparameters (they did not report hyperparameters). While BART (Lewis et al., 2020) and T5 (Raffel et al., 2019) are state-of-the-art in SciGen (Moosavi et al., 2021), MBD (Rebuffel et al., 2021) is the state-of-the-art in WikiBio (Lebret et al., 2016). It is a multi-branch decoder that was build to reduce the hallucination in data-to-text tasks.

E Ablation Study – Downstream Tasks

This section shows the results of our downstream ablation experiments. For experiments, we use the same setup as described in Section 6, i.e., we consider the large variant of BART (Lewis et al., 2020) with its default tokenisation (DT) and masking procedure (DM) as baseline for this experiment. Additionally, we finetune the models in the downstream task (using the hyperparameters described in Appendix A). For evaluation, we use the respective test splits (in-domain in case of InfoTabs (Gupta et al., 2020)). Table 14 and Table 15 show the results of our ablation experiments in downstream tasks. We conduct the same experiments as for the general ablation study (Section 6): *DT + INP* uses the default tokenisation but our masking procedure (the Inferable Number Prediction Task, Section 3.2), *CLT + INP* uses the character-level tokenisation for numbers (CLT), *Ours* combines CLT and INP with the contrastive loss (CL), and *Ours - INP* combines CLT with the contrastive loss but uses DM instead of INP. Overall, the results reflect the findings described in Section 6. We highlight statistically significant improvements of our approach over the respective baseline in the tables (independent two-sample t-test, $p \leq 0.05$).

	MoverScore	**BLEU**
	WikiBio	
BART	61.50	17.98
DT + INP	61.74	17.31
CLT + INP	**62.01**	**18.42**
Ours	**62.78**	**18.54**
Ours - INP	**62.15**	**18.25**
	SciGen	
BART	53.43	4.87
DT + INP	53.76	4.65
CLT + INP	**54.12**	**6.45**
Ours	**55.00**	**9.30**
Ours - INP	**54.87**	**7.32**

Table 14: Downstream ablation study for SciGen and WikiBio.

According to automatic metrics, the impact on table-to-text generation is rather limited. We suspect that this is partly due to their shortcomings in assessing the correctness of information not directly included in the source data (see also Section 5.1). DT + INP shows that pretraining using our masking procedure slightly improves the results in both cases. Using the character-level tokenisation for numbers further improves the results (CLT + INP). In case of SciGen, the comparison between Ours and Ours - INP suggests that using the character-level tokenisation and contrastive learning to improve the number representation has more impact than pretraining using INP. In case of WikiBio, the differences are rather negligible (although Ours outperforms the baseline). This might be due to the characteristics of the dataset. As described in Section 4.1, WikiBio rather requires copying numbers from input tables to output text, than inferring context-related numbers (which is different in the other datasets).

	EM	F1
DROP		
BART	36.00	39.26
DT + INP	**39.87**	**43.77**
CLT + INP	**42.19**	**46.09**
Ours	**45.60**	**49.50**
Ours - INP	**43.68**	**47.45**
InfoTabs		
BART	33.30	-
DT + INP	**48.21**	-
CLT + INP	**61.56**	-
Ours	**67.20**	-
Ours - INP	**62.56**	-

Table 15: Downstream ablation study for DROP and InfoTabs

In case of DROP (Dua et al., 2019) and InfoTabs (Gupta et al., 2020), the results are more expressive. In both cases, we find that just using INP (DT + INP) as an extended pretraining task already brings a significant improvement over the baselines. This is further improved by using character-level tokenisation for numbers (CLT + INP) and contrastive learning (Ours). Ours - INP shows that in both cases, INP has a significant impact on performance improvements.

F 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 using the Inferable Number Prediction Task. We focus on BART (Lewis et al., 2020) (the large variant) for this experiment. We conduct this experiment using the large split of the SciGen dataset (Moosavi et al., 2021). Table 16 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 16: Comparison of results when using different representations for incorporating the character-level tokenisation.

G 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 17 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. We highlight statistically significant improvements of our approach over the respective baseline in the tables (independent two-sample t-test, $p \leq 0.05$).

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

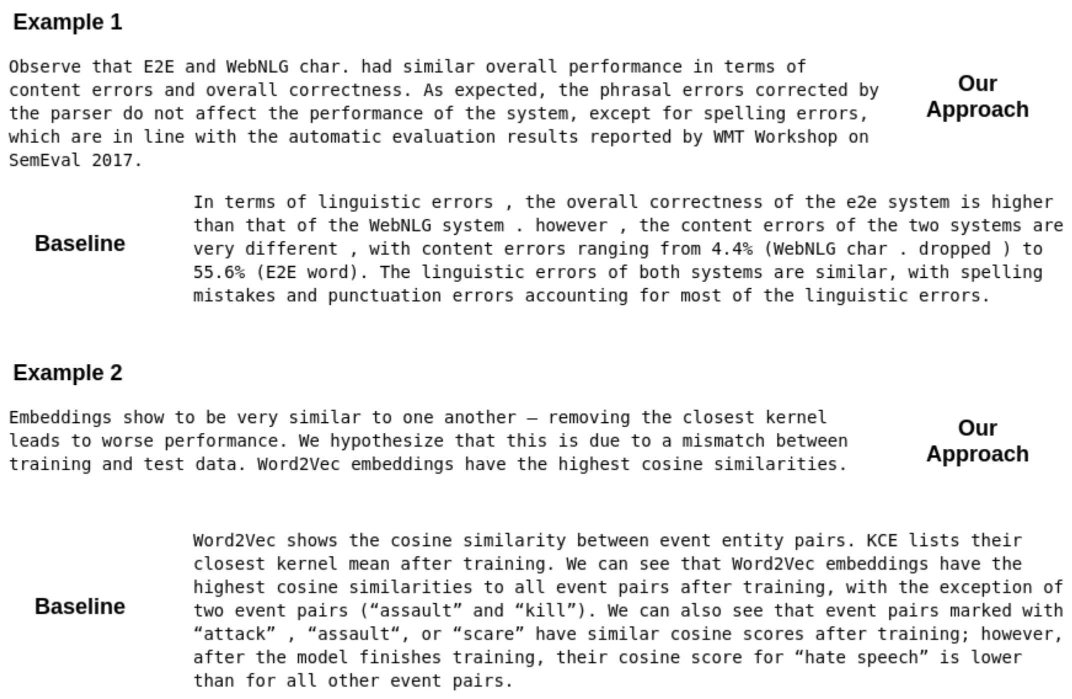
Baseline refers to the BART-large model. *MWP-pretrained Baseline* shows the results for Baseline, but further pretrained on MWP. *MWP-pretrained Baseline + CLT* represents the results for the MWP-pretrained Baseline, but uses the character-level representation (CLT) for numbers instead of BART's default 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.

H 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.



Example 1

Observe that E2E and WebNLG char. had similar overall performance in terms of content errors and overall correctness. As expected, the phrasal errors corrected by the parser do not affect the performance of the system, except for spelling errors, which are in line with the automatic evaluation results reported by WMT Workshop on SemEval 2017.

Our Approach

Baseline

In terms of linguistic errors , the overall correctness of the e2e system is higher than that of the WebNLG system . however , the content errors of the two systems are very different , with content errors ranging from 4.4% (WebNLG char . dropped) to 55.6% (E2E word). The linguistic errors of both systems are similar, with spelling mistakes and punctuation errors accounting for most of the linguistic errors.

Example 2

Embeddings show to be very similar to one another — removing the closest kernel leads to worse performance. We hypothesize that this is due to a mismatch between training and test data. Word2Vec embeddings have the highest cosine similarities.

Our Approach

Baseline

Word2Vec shows the cosine similarity between event entity pairs. KCE lists their closest kernel mean after training. We can see that Word2Vec embeddings have the highest cosine similarities to all event pairs after training, with the exception of two event pairs (“assault” and “kill”). We can also see that event pairs marked with “attack” , “assault“, or “scare” have similar cosine scores after training; however, after the model finishes training, their cosine score for “hate speech” is lower than for all other event pairs.

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