An Empirical Investigation of Contextualized Number Prediction

Daniel Spokoyny
Carnegie Mellon University
dspokoyn@cs.cmu.edu

Taylor Berg-Kirkpatrick
UC San Diego
tberg@ucsd.edu

Abstract

We conduct a large scale empirical investigation of contextualized number prediction in running text. Specifically, we consider two tasks: (1) *masked number prediction* – predicting a missing numerical value within a sentence, and (2) *numerical anomaly detection* – detecting an errorful numeric value within a sentence. We experiment with novel combinations of contextual encoders and output distributions over the real number line. Specifically, we introduce a suite of output distribution parameterizations that incorporate latent variables to add expressivity and better fit the natural distribution of numeric values in running text, and combine them with both recurrent and transformer-based encoder architectures. We evaluate these models on two numeric datasets in the financial and scientific domain. Our findings show that output distributions that incorporate discrete latent variables and allow for multiple modes outperform simple flow-based counterparts on all datasets, yielding more accurate numerical prediction and anomaly detection. We also show that our models effectively utilize textual context and benefit from general-purpose unsupervised pretraining.[1]

1 Introduction

Pretraining large neural architectures (e.g. transformers (Devlin et al., 2019; Raffel et al., 2019)) on vast amounts of unlabeled data has lead to great improvements on a variety of NLP tasks. Typically, such models are trained using a masked language modeling (MLM) objective and the resulting contextualized representations are finetuned for a particular downstream task like question answering or sentence classification (Devlin et al., 2019; Lan et al., 2020). In this paper, we focus on a related modeling paradigm, but a different task. Specifically, we investigate contextualized number prediction: predicting a real numeric value from its textual context using an MLM-style modeling objective. We conduct experiments on two specific variants: (1) *masked number prediction* (MNM), in which the goal is to predict the value of a masked number token in a sentence, and (2) *numerical anomaly detection* (NAD), with the goal of deciding whether a specific numeric value in a sentence is errorful or anomalous. In contrast with more standard MLM training setups, here we specifically care about the accuracy of the trained masked conditional distributions rather than the contextualized representations they induce. While successful models for these tasks are themselves useful in applications like typo correction and forgery detection (Chen et al., 2019), better models of numeracy are essential for further improving downstream tasks like question answering, numerical information extraction (Mirza et al., 2017; Saha et al., 2017) or numerical fact checking (Thorne and Vlachos, 2017), as well as for processing number-heavy domains like financial news, technical specifications, and scientific articles. Further, systems that detect anomalous numbers in text have applications in practical domains – for example, medicine (Thimbleby and Cairns, 2010) – where identification of numerical entry errors is critical.

Our modeling approach to contextualized number prediction combines two lines of past work. First, following Chen et al. (2019), we treat number prediction as a sentence-level MLM problem where only numerical quantities are masked. However, Chen et al. (2019) focused on predicting the discrete exponent of masked numbers as a classification problem. In contrast, Spithourakis and Riedel (2018) demonstrate the utility of predicting full numerical quantities in text, represented as real numbers, but do so in a language modeling frame-

[1] Code available at: https://github.com/dspoka/mnm

work, conditioned only on left context. Here, we propose a novel setup that combines full-context encoding (i.e. both left and right contexts) with real-valued output distributions for modeling numerical quantities in text. In Figure 1, we illustrate an example where we aim to predict “2 trillion” as a quantity on the real number line.

We expand upon past work by conducting a large scale empirical investigation that seeks to answer three questions: (1) Which encoding strategies yield more effective representations for numbers in surrounding context? (2) Which encoding architectures provide the best representations of surrounding context? (3) What are the most effective real-valued output distributions to model masked number quantities in text? To answer these questions, we propose a suite of novel real-valued output distributions that add flexibility through the use of learned transformation functions and discrete latent variables. We conduct experiments for both MNM and NAD tasks on two large datasets in different domains, combining output distributions with both recurrent and transformer-based encoder architectures, as well as different numeric token encoding schemes. Further, while Chen et al. (2019) studied a specific type of NAD (detecting exaggerated numbers in financial comments), we examine several NAD variants with different types of synthetic anomalies that are found to arise in practice across different domains of data. Finally, we further compare results with a strong discriminative baseline.

2 Models

Our goal is to predict numbers in their textual contexts. The way we approach this is similar to masked language modeling (MLM), but instead of masking and predicting all token types, we only mask and predict tokens that represent numeric values. For example in Figure 1 we wish to predict that the value of the masked number `[#MASK]` should be $2 \times 10^{12} \in \mathbb{R}$ given the surrounding context.

For notational simplicity, we describe our model as predicting a single missing numeric value in a single sentence. However, like other MLMs (see section 4.3), during training we will mask and predict multiple numeric values simultaneously. Let $\boldsymbol{X}$ be a sentence consisting of N tokens where the kth token is a missing numerical value, y. The goal of our model is to predict the value of y conditioned on $\boldsymbol{X}$. We will use common notation for from similar setups and simply treat the kth token in X as a masked numeric value, `[#MASK]`.

Our models $P_{\theta,\gamma}(y|\boldsymbol{X})$ consist of three main components: an input representation of the sentence, a contextual encoder with parameters γ which summarizes the sentence, and an output distribution with parameters θ over the real number line. In this section we will describe our strategies for numerical input representation, the two types of contextual encoders we use, along with different formulations of numerical output distributions.

2.1 Input Context Representation

We first describe the input representation for the textual context $\boldsymbol{X}$ that will be passed into our model's encoder. We let x_i represent the ith token in the input sequence. Like related MLMs that leverage transformers (which is one type of encoder we consider in experiments) we separate the representation of x_i into several types of embeddings. We include a positional embedding e^{POS} and a wordpiece token embedding e^{TOK} like the original BERT. We also introduce our new numeric value embedding e^{NUM} to help us learn better numerical representations. Finally, as shown in Figure 1, the input representation for token x_i is the sum of these three H-dimensional embeddings.

If the token at position i represents a numerical quantity, we replace it with a special symbol `[#MASK]`, and represent its numerical value using e_i^{NUM}.[2] We use the extraction rules detailed in Section 3.1 to find the numbers in our input sequence. In the next section we will describe two strategies for numerical representation e^{NUM}.

2.1.1 Digit-RNN Embedding

The large range ($[1, 1e^{16}]$ in our data) of numerical values prevents them from being used directly as inputs to neural network models as this results in optimization problems due to the different scales of parameters. One strategy to learn embeddings of numerical values has been shown by Saxton et al. (2019) which used character-based RNNs to perform arithmetic operations such as addition and multiplication. We conduct experiments with a similar strategy and represent each number in scientific notation (d.ddde+d) with 6 digits of precision as

[2] We exclude segment type embeddings since we do not perform next sentence prediction. We also found it helpful to use the zero vector as the numerical embedding for e_i^{NUM} if position i is not a quantity.

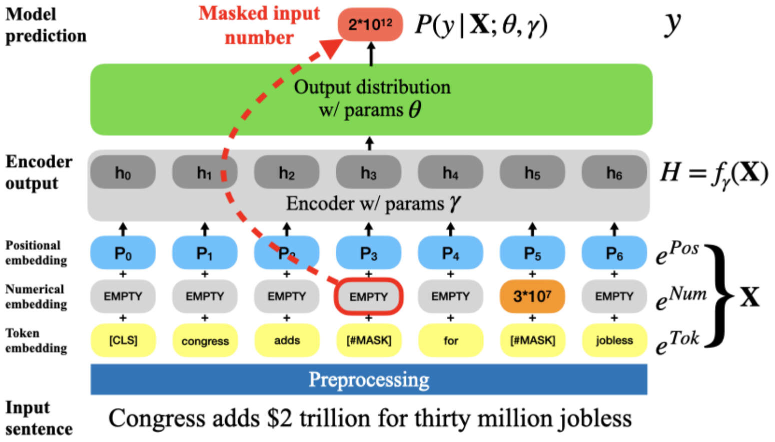

Figure 1: Outline of our model architecture consisting of a sentence representation $\boldsymbol{X}$ which is fed to the encoder with parameters γ and an output distribution over the real number line with parameters θ. In this example our masked numerical objective is to predict the masked out "2 trillion" quantity y. Note that our model is able to use a numerical embedding of the unmasked input $3 * 10^7$ value ("thirty million") as part of the context.

a string. We then use a digit-RNN to encode the string and use the last output as e^{NUM}.

2.1.2 Exponent Embedding

A simpler approach to represent numbers would be to explicitly learn embeddings for their magnitudes. Magnitudes have been shown to be a key component of the internal numerical representation of humans and animals (Ansari, 2016; Whalen et al., 1999; Dehaene et al., 1998). We conduct experiments with an encoding scheme that learns embeddings for base-10 exponents.

2.2 Context Encoder

The encoder's goal is to summarize the surrounding text, along with other numbers that appear therein. We define $\boldsymbol{H} = f_\gamma(\boldsymbol{X})$ where the encoder f_γ is a function of the context $\boldsymbol{X}$, and $\boldsymbol{H}$ is the hidden representation of the encoder's last layer. Next, we describe two encoder architectures: a transformer and a recurrent approach.

2.2.1 Transformer Encoder

Transformer architectures pretrained on vast amounts of data have led to breakthroughs in textual representation learning (Yang et al., 2019; Liu et al., 2019; Lan et al., 2020; Raffel et al., 2019). We use the 12-layer BERT-base architecture (Devlin et al., 2019) with the implementation provided by Huggingface (Wolf et al., 2019). We use the original BERT's word-piece vocabulary with 30,000 tokens and add a new `[#MASK]` token.

2.2.2 BiGru Encoder

Previous methods focusing on the related task of predicting the order of magnitude of a missing number in text showed that RNNs were strong models for this task (Chen et al., 2019). In our real-valued output task we use a bidirectional Gated Recurrent Unit (*BiGRU*), the best performing model from Chen et al. (2019). We use a one-layer BiGRU with a 64-dimensional hidden state and a dropout layer with a 0.3 dropout rate. We use the same pretrained word-piece embeddings from BERT as this allows us to directly compare the two encoders.

2.3 Real-valued Output Distributions

In early experiments, we observed that simple continuous distributions (e.g. Gaussian or Laplace) performed poorly. Since numbers can have ambiguous or underspecified units, and further, since numbers in text are heavy-tailed, asymmetric or multi-modal output distributions may be desirable. For this reason, we propose several more flexible output distributions, some which include learned transforms and others which include latent variables (both well-known methods for adding capacity to real-valued distributions), to parameterize $P(y|\boldsymbol{X})$.

2.3.1 Log Laplace

A common method for constructing expressive probability density functions is to pass a simple density through a transformation (e.g. a flow or invertible mapping function). As an initial example (and our first output distribution), we describe the log Laplace distribution as a type of flow. Since

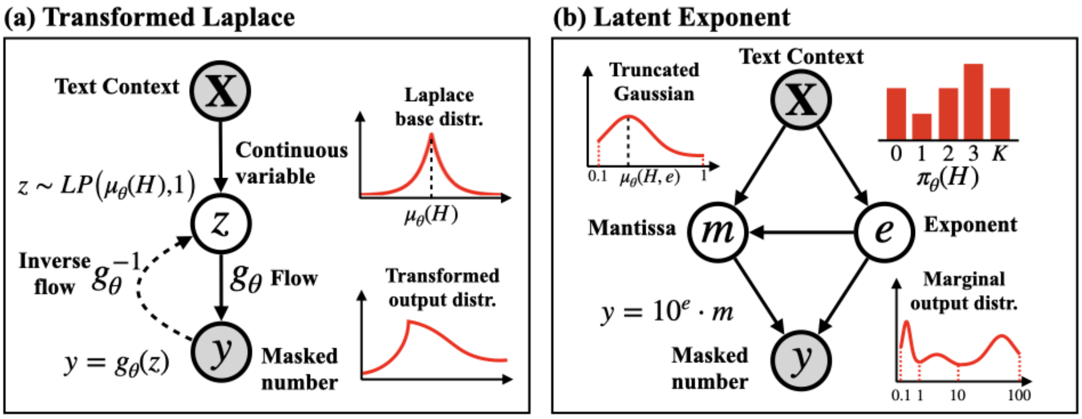

Figure 2: *Left (a):* We depict our *LogLP* and *FlowLP* graphical models along with the latent and output distributions. *Right (b):* Probabilistic graphical model of our latent *DExp* model.

numbers in text are not distributed evenly on the number line due to a long tail of high magnitudes, a simple trick is to instead model the log of numeric values. If the base distribution is Laplace, this yields a log Laplace distribution, which we describe next as an exponential transformation.

In Figure 2, we illustrate our *LogLP* model with a continuous intermediate variable z, encoder f_γ, with exp as the transformation, g_θ , and consequently log as g_θ^{-1}. In equation 1 we show our generative process and training objective where both g_θ and g_θ^{-1} are deterministic functions with no parameters. We let $\mu_\theta(\boldsymbol{H})$ denote a single layer MLP that outputs the location parameter of the base Laplace distribution on z, which is transformed to produce the output variable, y. More precisely:

Generative Process:

$$z \sim Laplace(\mu_\theta(H), 1)$$

$$y = g_\theta(z) = \exp z$$

Training Objective:

$$g_\theta^{-1}(y) = \log y$$

$$\log P(y \,|\, \mathbf{X}) = -\left| g_\theta^{-1}(y) - \mu_\theta(H) \right| - C + \log J_{det}(y)$$

$$\log J_{det}(y) = \log \left| \frac{dg^{-1}}{dy}(y) \right| = \log \left| \frac{1}{y} \right| \qquad (1)$$

2.3.2 Flow-transformed Laplace

The exp transformation may not be the ideal choice for our data. For this reason we consider a parameterized transform (flow) to add further capacity to the model. For our purposes, we are restricted to 1-dimensional transformations $g : \mathbb{R} \to \mathbb{R}$. Further, by restricting the class of functions, we ensure an efficient way of computing the log-derivative of the inverse flow, which allows us to efficiently compute likelihood. We conduct experiments with the simple parameterized flow described in Equation 2. We use a single layer MLP to independently predict each parameter a,b,c from $\boldsymbol{H}$, the output of $f_\gamma(\boldsymbol{X})$. We also scale the range of b, c to be between [0.1, 10] using a Sigmoid activation. Similarly to the *LogLP* setting, $\mu_\theta(\boldsymbol{H})$ is a single layer MLP which predicts the location parameter of the Laplace.

Generative Process:

$$z \sim Laplace(\mu_\theta(H), 1)$$

$$y = g_\theta(z) = \frac{\exp(\frac{z-a}{b})}{c}$$

Training Objective:

$$g_\theta^{-1}(y) = a + b \log cy$$

$$\log P(y \,|\, \mathbf{X}) = -\left| g_\theta^{-1}(y) - \mu_\theta(H) \right| - C + \log J_{det}(y)$$

$$\log J_{det}(y) = \log \left| \frac{dg^{-1}}{dy}(y) \right| = \log \left| \frac{b}{y} \right| \qquad (2)$$

This parameterization of flow is designed to allow for (1) re-centering of the input variable (via parameter a), (2) re-scaling of the input (via parameter b), and (3) re-scaling of the output (via parameter c). Together, this leads to a family of inverse flows that are all log-shaped (i.e. they compress higher values), yet have some flexibility to change intercept and range.

2.3.3 Discrete Latent Exponent

While *FlowLP* adds flexibility over the *LogLP* model, both have the drawback of only being able to produce unimodal output distributions.[3] A well-established approach to parameterizing multimodal densities is to use a mixture model. The mixture component is determined by a discrete latent

[3] In principle, more complicated flows could also have multiple modes – though they are more challenging to construct and optimize.

variable in contrast with the continuous intermediate variable introduced in the flow-based models. In Figure 2 we show our *DExp* model where e represents an exponent sampled from a multinomial distribution, and m is the mantissa sampled from a truncated Gaussian.

Prior work has shown the effectiveness of cross-entropy losses on numerical training (Saxton et al., 2019; Chen et al., 2019). For this reason we use a truncated Gaussian on the range of [0.1,1] to generate m, which effectively restricts back-propagation to a single mixture component for a given observation. The combination of exponent and mantissa prediction allows us to benefit from the effectiveness of cross-entropy losses, while at the same time getting more fine-grained signal from the mantissa loss. In Equation 3 we show the *DExp* generative process and training objective. We let $\pi_\theta(\boldsymbol{H})$ denote a single layer MLP that outputs the multinomial parameters of $P(e|X)$. Similarly, we let $\mu_\theta(\boldsymbol{H}, e)$ denote a two layer MLP with a [.1,1] scaled Sigmoid that outputs the mean parameter of the mantissa normal distribution.

Generative Process:

$$e \sim Mult(\pi_\theta(H))$$

$$y \sim \mathcal{N}_{trunk[0.1,1]}(\mu_\theta(H, e), 0.05)$$

Training Objective:

$$e*(y) = \lfloor log_{10}(y) \rfloor$$

$$\log P(y\,|\,\mathbf{X}) = \log \Big[P\big(e = e*(y)\big) \cdot \frac{1}{C} \exp\Big(-10\Big(\frac{y}{10^{e*(y)}} - \mu_\theta\big(H, e*(y)\big) \Big)^2 \Big) \Big] \tag{3}$$

2.3.4 Gaussian Mixture Model

Inspired by the best performing model from Spithourakis and Riedel (2018) we also compare with a Gaussian mixture model (*GMM*). This model assumes that numbers are sampled from a weighted mixture of *K* independent Gaussians. During training the mixture from which a particular point was sampled from is not observed and so it is treated as a latent variable. We can optimize the marginal log-likelihood objective by summing over the K mixtures. In equation 4, *GMM* has *K* mixtures parameterized by K means and variances $\boldsymbol{\mu}, \boldsymbol{\sigma}$, respectively. Following Spithourakis and Riedel (2018), we pre-train the parameters $\boldsymbol{\mu}, \boldsymbol{\sigma}$ on all the numbers in our training data $\mathcal{D}$ using EM. The means and variances are then fixed and our masked number prediction model only predicts mixture weights during training and inference. We let $\pi_\theta(\boldsymbol{H})$ denote a single layer MLP that outputs the mixture weights $P(e|X)$.

Generative Process:

$$\mu = [\mu_1, \mu_2, \ldots \mu_k]; \sigma = [\sigma_1, \sigma_2, \ldots \sigma_k]$$

$$e \sim Mult(\pi_\theta(H))$$

$$y \sim \mathcal{N}(\mu_e, \sigma_e)$$

Training Objective:

$$\log P(y\,|\,\mathbf{X}) = \log \sum_{k=1}^{K} \Big[P(e = k) \cdot \frac{1}{C} \exp\Big(\frac{-(y - \mu_k)^2}{2\sigma_k^2} \Big) \Big] \tag{4}$$

3 Data

Financial news Financial news documents are filled with many different ratios, quantities and percentages which make this domain an ideal test-bed for MNM. The **FinNews** is a collection of 306,065 financial news and blog articles from websites like Reuters[4]. We randomly break the documents into [train, valid, test] splits with [246065, 30000, 30000] respectively.

Since **FinNews** has many occurrences of dates and years, we also evaluate on a subset corpus, **FinNews-$** , to measure effectiveness at modeling only dollar quantities in text. **FinNews-$** is constructed exactly as **FinNews** , with the added requirement that the number is preceded by a dollar sign token ($). For all training and testing on **FinNews-$** , we only predict dollar values.

Academic papers Academic papers have diverse semantic quantities and measurements that make them an interesting challenge numeracy modeling. For this reason, we also use S2ORC, a newly constructed dataset of academic papers (Lo et al., 2020). We use the first 24,000 full text articles, randomly splitting into [20000, 2000, 2000] [train, valid, test] splits. [5] We refer to this dataset as **Sci**. All three datasets follow the same preprocessing discussed below and summary statistics are provided in Table 1.

3.1 Preprocessing

Financial news, academic papers, and Wikipedia articles all have different style-guides that dictate how many digits of precision to use or whether certain quantities should be written out as words. While such stylistic queues might aid models in better predicting masked number *strings*, we are

[4] www.kaggle.com/jeet2016/us-financial-news-articles

[5] We also filter articles from only these categories {Geology, Medicine, Biology, Chemistry, Engineering, Physics, Computer science, Materials science, Economics, Business, Environmental science}.

specifically focused on modeling actual numeric values for two reasons: (1) reduced dependence on stylistic features of the text domain leads to better generalization to new domains, and (2) the numerical value of a numeric token conveys its underlying meaning and provides a finer-grained learning signal. For example currencies are usually written as a number and magnitude like *$32 million* however, many quantities can be written out as cardinals *sixty thousand trucks*. We normalize our input numbers so that changing the style from *five* to *5* does not change our output predictions.

As exemplified in Figure 1, the aim of our approach is to incorporate both numbers as context and numbers as predictions (i.e. 2 trillion and thirty million in the example). For this reason, before tokenization we employ heuristics to combine numerals, cardinals and magnitudes into numerical values, whilst removing their string components. We also use heuristics to change ordinals into numbers. By following this normalization preprocessing procedure we get higher diversity of naturally occurring quantitative data and mitigate the bias towards some particular style guide.

For both **FinNews** and **Sci** we lowercase the text and ignore signs $(+, -)$, so all numbers are positive and restrict magnitudes to be in $[1, 1e^{16}]$. We discard sentences that do not have numbers or where the numbers are outside of our specified range. We also filter out sentences that have less than eight words and break up sentences longer than 50 words.[6] We do not use the special token *[SEP]* and all examples are truncated to a maximum length of 128 tokens.

4 Experiments

In this section we explain our experimental setup, starting with our evaluation metrics, implementation details, results, and ablation analyses. We use the following naming convention for models: we specify the encoder (*BiGRU*, *BERT*) first, followed by one of our four output distributions (*LogLP*, *FlowLP*, *DExp*, *GMM*).

4.1 Evaluation

For the MNM task on $\mathcal{D}_{\text{valid}}$ and $\mathcal{D}_{\text{test}}$ splits we randomly select a single number to mask out from the input and predict. We let $\hat{y}$ denote the model's $\arg\max$ prediction from $P(y|\boldsymbol{X})$ and y as the actual observed number. In equation 5 and 6 we show how we calculate log-MAE (*LMAE*) and exponent accuracy (*E-Acc*), both of which use log base 10.

$$\textit{LMAE} \;=\frac{1}{|\mathcal{D}_{\text{test}}|}\sum_{\mathcal{D}_{\text{test}}}|\log y-\log\hat{y}| \tag{5}$$

$$\textit{E-Acc} \;=\frac{1}{|\mathcal{D}_{\text{test}}|}\sum_{\mathcal{D}_{\text{test}}}\mathbb{1}[\lfloor\log y\rfloor=\lfloor\log\hat{y}\rfloor] \tag{6}$$

4.2 Numerical Anomaly Detection

Both *LMAE* and *E-Acc* metrics test the model's argmax prediction and not the entire $P(y|\boldsymbol{X})$ distribution. We next consider the NAD task where our models need to discern the true number versus some anomaly. We let $\tilde{y}$ denote an anomaly and describe two different ways, [*string, random*], we construct an anomalous example. For *string* we use the true y and randomly perform one of three operations [*add, del, swap*]: inserting a new digit, deleting an existing digit, and swapping the first two digits respectively. For *random*, we randomly sample a number from the training data $\mathcal{D}$ as our anomaly. We choose these string functions as they constitute a large part of numerical entry errors (Thimbleby and Cairns, 2010; Wiseman et al., 2011). Further, *random* mimics a copy-paste error. We report the AUC of a ROC curve for both types as random-anomaly (*R-AUC*) and string-anomaly (*S-AUC*) respectively, using the model's output density to rank the true value against the anomaly.

4.3 Implementation Details

We train all models with stochastic gradient descent using a batch-size of 32 for 10 epochs. We use early stopping with a patience of three on the validation loss. For pretrained *BERT* encoder experiments, we use two learning rates $\{3e^{-5}, 1e^{-2}\}$ for all pretrained parameters and newly added parameters respectively. For all non-pretrained *BERT* experiments and all *BiGRU* encoders we use a single learning rate of $2e^{-2}$.

Devlin et al. (2019) propose a two step process to generate masked tokens. First, select tokens for masking with an independent probability of 15%. Second, for a selected token: With 80% probability replace it with a `[MASK]`, 10% replace it with a random token, and 10% leave it unchanged. Since there are fewer numbers than text tokens, we use a higher probability of 50% for selection. We follow a similar strategy for masking numbers: 80% of the time masking out the number, 10% of the time

[6] Sentences under eight words in length tended to be titles of articles with the date as the only numeric quantity.

	FinNews			**FinNews-$**			**Sci**		
	train	**valid**	**test**	**train**	**valid**	**test**	**train**	**valid**	**test**
#instances	522996	58095	64433	188286	22338	23281	360514	36523	36104
avg-length	102.5	108.3	108.9	115.2	115.4	116.1	125.6	126.4	126.5
%numbers	8.8	9.3	9.6	13.0	12.7	13.2	7.1	7.2	7.1
min	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
median 50	313.0	250.0	329.0	2016.0	2016.0	2016.0	9.0	8.0	9.0
median 75	3141.0	2558.0	3500.0	$\sim 10^{4}$	$\sim 10^{4}$	$\sim 10^{4}$	42.0	40.0	43.0
median 90	$\sim 10^{6}$	$\sim 10^{6}$	$\sim 10^{6}$	$\sim 10^{7}$	$\sim 10^{7}$	$\sim 10^{7}$	1959.0	1948.0	1972.1
max	$\sim 10^{15}$	$\sim 10^{14}$	$\sim 10^{15}$	$\sim 10^{15}$	$\sim 10^{14}$	$\sim 10^{15}$	$\sim 10^{15}$	$\sim 10^{14}$	$\sim 10^{15}$

Table 1: Statistics on our datasets. The top half of the table reveals the number of examples per data split, the average length of sentences, and the fraction of tokens that are numbers. The bottom half shows summary statistics for number values in both datasets.

randomly substituting it with a number from train, and 10% of the time leaving it unchanged.

Baselines: We also consider a fully discriminative baseline trained to predict real vs. fake numbers with binary cross entropy loss. The negative numerical samples are randomly drawn from training set numbers to match exactly the random-anomaly task. During training each positive datum has one negative example and is trained in the same batch-wise fashion. When this model uses exponent embeddings for output numbers, *emb*$_{exp}$, we can also calculate the exponent accuracy by selecting the exponent embedding with highest model score as a predicted value. We include this approach in experiments as a non-probabilistic alternative to our four output distributions.

4.4 Results

We ran all combinations of encoders and output distributions using input exponent embeddings on **FinNews** and show the results in Table 2. We train the *GMM* model with four different settings of K $\in \{31, 63, 127, 255\}$ and report results for the highest-performing setting.

Comparing the two encoders, we find that *BERT* results in stronger performance across all metrics and all output distributions. Although both settings share the same pretrained embedding layers, the pretrained transformer architecture has higher capacity and is able to extract more relevant numerical information for both MNM and NAD.

We find that the parameterized *FlowLP* model was generally better across all metrics under both encoders compared to the *LogLP* model. With the weaker *BiGRU* encoder, the *LogLP* model's *S-AUC* is only 0.04 better than random guessing.

The *DExp* model was the best performing output distribution across all metrics and both encoders, yielding on average 10% higher *E-Acc* and a gain of 0.13 on *AUC*. This means that *DExp* had the best overall fit in terms of the predicted mode $(\arg\max)$ as well as the overall density $P(y|\boldsymbol{X})$.

In contrast, *GMM* , which is also a discrete latent variable model capable of outputting a multi-modal distribution, underperformed across all metrics. There was little effect from adjusting the number of mixture components, with slight improvements using more mixtures. One possible reason for the *GMM* model's worse performance is that the mixtures are fit and fixed before training without any of the surrounding textual information. Quantities such as dates and years have many textual clues, but the model's initial clustering may group them together with other quantities. We also found that, empirically, optimization for this model was somewhat unstable.

Finally the *Disc* baseline was the second best performing model on NAD , though on MNM it showed worse *E-Acc* than *LogLP* and *FlowLP* models. This baseline benefited from being directly trained for NAD , which may explain it's underperformance on MNM metrics. Due to the comparatively worse performance of both the *BiGRU* encoder and the *GMM* output distribution, we exclude them from the remainder of our experiments.

4.5 Ablations

Ablations on Numerical Embedding We select our best performing model, *BERT-DExp*, and ablate the numerical input representation on **FinNews**. We compare using *emb*$_{dig}$, *emb*$_{exp}$, and a version of *ExpBert* which has no numerical input representation. The top half of Table 3 displays the results.

Model	**LMAE**↓	***E-Acc*** ↑	**r-AUC**↑	**s-AUC**↑
Train-Mean	7.69	1.03	-	-
Train-Median	1.88	5.52	-	-
BiGRU-Disc	-	55.8	0.756	0.646
BiGRU-LogLP	0.671	58.8	0.675	0.548
BiGRU-FlowLP	0.622	61.8	0.694	0.591
BiGRU-DExp	0.576	71.5	0.843	0.821
BERT-Disc	-	62.7	0.762	0.656
BERT-GMM K=255	1.18	21.3	0.585	0.440
BERT-LogLP	0.5666	64.9	0.686	0.557
BERT-FlowLP	0.5732	65.5	0.717	0.609
BERT-DExp	**0.500**	**74.6**	**0.861**	**0.828**

Table 2: Results on **FinNews** where all models use input exponent embeddings emb_{exp} and all *BERT* encoders are pretrained. We also include the mean and median number from training $\mathcal{D}$ as simple baselines.

Ablation Type	**LMAE**↓	***E-Acc*** ↑	**r-AUC**↑	**s-AUC**↑	**all-LMAE**↓	**all-*E-Acc*** ↑
Numerical Input Embedding						
BERT-DExp (All #'s Masked)	0.656	66.5	0.831	0.809	0.656	66.5
BERT-DExp + emb_{exp}	0.500	74.6	0.861	0.828	0.888	62.2
BERT-DExp + emb_{dig}	0.506	74.4	0.858	0.826	0.920	62.1
BERT-DExp + emb_{exp} + emb_{dig}	0.498	74.9	0.861	0.828	0.899	62.3
No Pretraining						
BERT-DExp + emb_{exp}	0.615	68.8	0.840	0.810	0.889	60.6
BERT-FlowLP + emb_{exp}	0.769	57.9	0.670	0.563	0.861	54.4
BERT-Disc + emb_{exp}	-	26.9	0.632	0.599	-	-
BERT-LogLP + emb_{exp}	0.630	63.2	0.678	0.550	0.850	57.1

Table 3: Ablation on **FinNews** dataset. The top half of the table shows the effect of the numerical input representation. The bottom half shows performance for models trained from scratch, without leveraging pretrained BERT parameters.

We see that emb_{dig} and emb_{exp} perform equally well. Using no input number embeddings reduces performance by 8% on *E-Acc* and 0.03 *AUC* on both anomaly metrics. We also see that there is no benefit from combining both of these input representations, which implies that the model is able to extract similar information from each.

Ablations One-vs-All To measure our model's effectiveness at using the other numbers in the input we construct an ablated evaluation *All* , where all input numbers are masked out.[7] In Table 3 we see that all models that have a numerical embedding suffer a performance drop of around 12% *E-Acc* and an increase of 0.4 on *LMAE*. This suggests that the model is in fact using the other quantities for its predictions. We also find that the model with no input number embeddings does better on the *All* setting since it was effectively trained with fully masked input numbers.

Ablations on Pretraining In the bottom half of Table 3, we compare the effect of starting from a pretrained transformer versus training from scratch. We see that training from scratch hurts all models by around 6% on *E-Acc* and 0.02 on *R-AUC*. We also note that *BERT-LogLP* seems least affected, dropping only 1% on *E-Acc*.

Modeling Additional Domains In this section we explore how different models behave on the alternative domain of academic papers, and how modeling is affected by focusing only dollar quantities in financial news. In Table 4, we show results for pretrained BERT encoder models with input exponent embeddings, trained and evaluated on **Sci** and **FinNews-$** datasets.

On the **Sci** data, the generative models have similar performance on *LMAE* and *E-Acc* . We further find that *BERT-DExp* is still the best performing model across most metrics on both **Sci** and **FinNews-$** data. The *BERT-Disc* baseline, which is directly trained to predict anomalies, is consistently the second best across all datasets on NAD. Finally, we find that the **FinNews-$** is the most challenging of the three datasets, with *BERT-DExp* dropping on *E-Acc* by 20% compared to **FinNews** data. This supports our initial reasoning that the distribution of dollar amounts is more difficult to characterize than other quantities, such as dates, which tend to cluster to smaller ranges.

[7] To make comparisons exact, every test example has at least 2 numerical values so that we can perform this ablation.

	FinNews-$				**Sci**			
Model	**LMAE**↓	***E-Acc*** ↑	**r-AUC**↑	**s-AUC**↑	**LMAE**↓	***E-Acc*** ↑	**r-AUC**↑	**s-AUC**↑
BERT-Disc	-	46.9	0.828	0.588	-	68.8	0.722	0.657
BERT-LogLP	1.04	43.6	0.641	0.528	**0.374**	78.2	0.624	0.609
BERT-DExp	**0.91**	**56.9**	**0.867**	**0.678**	0.385	**81.0**	**0.786**	**0.836**
BERT-FlowLP	1.11	39.3	0.538	0.518	**0.374**	77.6	0.658	0.672

Table 4: Results on **FinNews-$** and **Sci** where all models use input exponent embeddings emb_{exp} and all *BERT* encoders are pretrained.

5 Related Work

Math & Algebraic Word Problems: There is a wide literature on using machine learning to solve algebraic word problems (Ling et al., 2017; Roy and Roth, 2016; Zhang et al., 2019), building novel neural modules to directly learn numerical operations (Trask et al., 2018; Madsen and Johansen, 2020) and solving a variety of challenging mathematical problems (Saxton et al., 2019; Lee et al., 2020; Lample and Charton, 2020). In these tasks, numbers can be treated as symbolic variables and computation based on these values leverages a latent tree of arithmetic operations. This differs from our task setting since there is no "true" latent computation that generates all the quantities in our text given the available context.

Numerical Question Answering The DROP dataset (Dua et al., 2019) is a new dataset that requires performing discrete numerical reasoning within a traditional question answering framework. Andor et al. (2019) treat DROP as a supervised classification problem, while recent work by Geva et al. (2020) show how synthetic mathematical training data can build better numerical representations for DROP. Unlike work on DROP, our primary focus is on the task of contextualized number prediction and numerical anomaly detection in text, which involve correlative predictions based on lexical context rather than concrete computation.

String Embeddings Recently, word and token embeddings have been analyzed to see if they record numerical properties (for example, magnitude or sorting order) (Wallace et al., 2019; Naik et al., 2019). This work finds evidence that common embedding approaches are unable to generalize to large numeric ranges, but that character-based embeddings fare better than the rest. However, this line of work also found mixed results on overall numeracy of existing embedding methods and further investigation is required.

Numerical Prediction Spithourakis and Riedel (2018) trained left-to-right language models for modeling quantities in text as tokens, digits, and real numbers using a *GMM*. Our empirical investigation focuses on MNM and considers both left and right contexts of numbers, along with a broader class of generative output distributions. Chen et al. (2019) predict magnitudes of numbers in text and also consider a type of NAD to detect numerical exaggerations on financial data. However, this modeling approach is restricted: it can only distinguish anomalies that result in a change of exponent. In contrast, our real-valued distributions allow us to focus on a broader suite of harder anomaly detection tasks, such as random substitutions and string input error.

6 Conclusion

In this work we carried out a large scale empirical investigation of masked number prediction and numerical anomaly detection in text. We showed that using the base-10 exponent as a discrete latent variable outperformed all other competitive models. Specifically, we found that learning the exponent representation using pretrained transformers that can incorporate left and right contexts, combined with discrete latent variable output distributions, results is the most effective way to model masked number quantities in text. Future work might explore combining more expressive flows with discrete latent variables.

Acknowledgements

We thank Volkan Cirik and the anonymous conference reviewers for providing valuable feedback. This project is funded in part by the NSF under grants 1618044 and 1936155, and by the NEH under grant HAA256044-17. The first author is supported in part by an NSF GRFP. Findings and observations do not necessarily reflect the views of funding agencies.

References

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

Daniel Ansari. 2016. Number symbols in the brain. In *Development of Mathematical Cognition*, pages 27–50. Elsevier.

Chung-Chi Chen, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen. 2019. Numeracy-600K: Learning numeracy for detecting exaggerated information in market comments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6307–6313, Florence, Italy. Association for Computational Linguistics.

Stanislas Dehaene, Ghislaine Dehaene-Lambertz, and Laurent Cohen. 1998. Abstract representations of numbers in the animal and human brain. *Trends in Neurosciences*, 21:355–361.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *ACL*.

Guillaume Lample and François Charton. 2020. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942.

Dennis Lee, Christian Szegedy, Markus Rabe, Sarah Loos, and Kshitij Bansal. 2020. Mathematical reasoning in latent space. In *International Conference on Learning Representations*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

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. *arXiv preprint arXiv:1907.11692*.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel S. Weld. 2020. S2ORC: The Semantic Scholar Open Research Corpus. In *Proceedings of ACL*.

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

Paramita Mirza, Simon Razniewski, Fariz Darari, and Gerhard Weikum. 2017. Cardinal virtues: Extracting relation cardinalities from text. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 347–351, Vancouver, Canada. Association for Computational Linguistics.

Aakanksha Naik, Abhilasha Ravichander, Carolyn Rose, and Eduard Hovy. 2019. Exploring numeracy in word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3374–3380, Florence, Italy. Association for Computational Linguistics.

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 e-prints*.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*.

Swarnadeep Saha, Harinder Pal, and Mausam. 2017. Bootstrapping for numerical open IE. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 317–323, Vancouver, Canada. Association for Computational Linguistics.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*.

Georgios P Spithourakis and Sebastian Riedel. 2018. Numeracy for language models: Evaluating and improving their ability to predict numbers. *arXiv preprint arXiv:1805.08154*.

Harold Thimbleby and Paul Cairns. 2010. Reducing number entry errors: solving a widespread, serious problem. *Journal of the Royal Society Interface*, 7(51):1429–1439.

James Thorne and Andreas Vlachos. 2017. An extensible framework for verification of numerical claims. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 37–40, Valencia, Spain. Association for Computational Linguistics.

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

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do nlp models know numbers? probing numeracy in embeddings. In *Empirical Methods in Natural Language Processing*.

John Whalen, Charles R Gallistel, and Rochel Gelman. 1999. Nonverbal counting in humans: The psychophysics of number representation. *Psychological science*, 10(2):130–137.

Sarah Wiseman, Paul Cairns, and Anna Cox. 2011. A taxonomy of number entry error. In *Proceedings of HCI 2011 The 25th BCS Conference on Human Computer Interaction 25*, pages 187–196.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Dongxiang Zhang, Lei Wang, Luming Zhang, Bing Tian Dai, and Heng Tao Shen. 2019. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE transactions on pattern analysis and machine intelligence*.