Improving Downstream Task Performance by Treating Numbers as Entities

Dhanasekar Sundararaman
Duke University
Durham, USA
dhanasekar.sundararaman@duke.edu

Vivek Subramanian
Amazon Alexa AI
Boston, USA
viveksub@amazon.com

Guoyin Wang
Amazon Alexa AI
Seattle, USA
guoyiwan@amazon.com

Liyan Xu
Emory University
Atlanta, USA
leonxu624@gmail.com

Lawrence Carin
Duke University
Durham, USA
lcarin@duke.edu

ABSTRACT
Numbers are essential components of text, like any other word tokens, from which natural language processing (NLP) models are built and deployed. Though numbers are typically not accounted for distinctly in most NLP tasks, there is still an underlying amount of numeracy already exhibited by NLP models. For instance, in named entity recognition (NER), numbers are not treated as an entity with distinct tags. In this work, we attempt to tap the potential of state-of-the-art language models and transfer their ability to boost performance in related downstream tasks dealing with numbers. Our proposed classification of numbers into entities helps NLP models perform well on several tasks, including a handcrafted Fill-In-The-Blank (FITB) task and on question answering, using joint embeddings, outperforming the BERT and RoBERTa baseline classification.

CCS CONCEPTS
• Computing methodologies → Natural language processing.

KEYWORDS
named entity, numeracy, text tagging

1 INTRODUCTION
Named entity recognition (NER) is the task of classifying nouns within a document into categories to which they belong [8]. Deep learning approaches for NER have been successful recently because of their ability to understand the underlying semantics of sentences [10] and incorporate this knowledge into tagging. Deep architectures for NER [4] include recurrent neural networks and attention based models including the Transformer [17].

While NER has been performed on a number of datasets, tagging types of numbers, or number entities, has not been a principal focus of NER [15]. This is despite the fact that numbers are prevalent in most texts, comprising roughly 5% of tokens in a sentence [6]. Numbers can take on several categories, including years, ages, phone numbers, dates, etc. Figure 1 shows sample sentences in which numbers are classified into entity types. Distinguishing these different types of numbers can greatly enhance performance on downstream tasks [11, 12].

In this paper, we present several methods for addressing number entity recognition (NuER). We begin by demonstrating that state-of-the-art NLP models are able to classify types of numbers that appear in text. We study this by utilizing BERT [3] to perform NuER on a custom version of the Stanford Question Answering Conference on Information and Knowledge Management (CIKM ’22), October 17–21, 2022, Atlanta, GA, USA. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557614

of their ability to understand the underlying semantics of sentences [10] and incorporate this knowledge into tagging. Deep architectures for NER [4] include recurrent neural networks and attention based models including the Transformer [17].

While NER has been performed on a number of datasets, tagging types of numbers, or number entities, has not been a principal focus of NER [15]. This is despite the fact that numbers are prevalent in most texts, comprising roughly 5% of tokens in a sentence [6]. Numbers can take on several categories, including years, ages, phone numbers, dates, etc. Figure 1 shows sample sentences in which numbers are classified into entity types. Distinguishing these different types of numbers can greatly enhance performance on downstream tasks [11, 12].

In this paper, we present several methods for addressing number entity recognition (NuER). We begin by demonstrating that state-of-the-art NLP models are able to classify types of numbers that appear in text. We study this by utilizing BERT [3] to perform NuER on a custom version of the Stanford Question Answering Dataset (SQuAD) [7], pruned to contain roughly 10K sentences for which the answers consist of numbers, and annotated in-house for numbers of six different entity types (year, percentage, date, count, age, and size) [11]. We hereby refer to this as the SQuAD-Num dataset and the BERT classifier used to predict entities as the NuER model. We discover that BERT model performs well at this task on most of the categories listed above. The success of this model is made possible by the highly contextualized embeddings (e.g., the appearance of the name of a month, say March, next to a year, say 2003) inferred by BERT.

We then verify the effectiveness of BERT fine-tuned on SQuAD-Num to tag numbers that appear in an out-of-domain, annotated, 2K-sentence subset of the Numeracy 600K dataset [2], which we call HA-Numeracy 2K. We chose this dataset because of the abundance and diversity of numbers present. The subset was sampled in such
Table 1: Number of entities in our annotation for each type in SQuAD-Num.

| Count | Size | Year | Percentage | Date | Age |
|-------|------|------|------------|------|-----|
| 1800  | 447  | 4355 | 291        | 418  | 82  |

a way as to maintain the distribution of number magnitudes that are originally present. We find that contextual embeddings are again beneficial in this setting, as surrounding tokens in both the source and target domains tend to be related. After verifying the quality of the annotations on the small sample, we annotated the full Numeracy 600K dataset using this model, which we hereafter call ENT-Numeracy 600K. Table 1 shows the number of samples in each entity type in SQuAD-Num. The annotations were conducted individually by three annotators and their common agreements were marked as labels. These annotations form the basis of all the models that we develop. We share these annotations with the NLP community to further enhance research in this direction.

Next, we jointly train embeddings for both tokens and number entities on the SQuAD-Num dataset. Answers to questions involving numbers are predicted by combining together token embeddings with trainable number entity embeddings. We find that this addition [13, 14], enables BERT and RoBERTa models to better answer questions in SQuAD involving numbers. We later analyze the effectiveness of NuER through a handcrafted Fill-In-The-Blank (FITB) classification task. Our proposed NuER model helps the models classify numbers better when presented with the entity type information.

In summary, our contributions are as follows:

1. We develop a BERT-based model for NuER which is used to tag numbers that appear in the SQuAD-Num dataset and in the out-of-domain HA-Numeracy 2K.
2. We release datasets containing number entity annotations for the NLP community.
3. We observe the effectiveness of NuER through a handcrafted Fill-In-The-Blank (FITB) task to predict a masked number in a sentence.
4. We jointly train embeddings for tokens and number entities, and show that numerical question answering performance improves when both are fed into the model.

2 RELATED WORK

Classification of numbers in plain text to entities has not been addressed methodologically and deserves considerable attention. [6] highlighted the importance of developing specialized embeddings for numbers such that basic properties including comparison and summation would be preserved. In addition, [11] designed a novel loss function that captured these properties and demonstrated improved performance on a variety of NLP tasks. Unlike their method which uses static embeddings, we propose to supplement the entity information of numbers in a contextual fashion to enable models to perform well on a number of tasks. Existing works have also shown that NLP models are intrinsically capable of numerical reasoning. This potential can be tapped using simple modifications like those proposed by [1], in which the BERT model was used to perform arithmetic operations. Like the above-mentioned works, we seek to leverage the potential of existing NLP models to classify numbers into different entity types and transfer such ability across related tasks [16] to boost performance.

3 PRELIMINARIES

For each corpus, we index sentences by \( n = 1, 2, \ldots, N \). Each token in a given vocabulary \( V \) is assigned a one-hot vector embedding \( v \in \{0, 1\}^{|V|} \). Tokens in a sequence are indexed by \( i = 1, 2, \ldots, T_n \), where \( T_n \) is the length of sequence \( n \) (after padding). During embedding lookup, tokens are mapped to embeddings \( e_{ni} \in \mathbb{R}^d \) (\( d \) refers to the dimension of embedding). For all tasks, we use pre-trained models including BERT and RoBERTa to form latent representations of inputs; token embeddings are summed with position embeddings \( p_{ni} \in \mathbb{R}^d \) and segment embeddings \( s_{ni} \in \mathbb{R}^d \) prior to being fed into the Transformer encoder. The output consists of a classification token \( e_n \in \mathbb{R}^d \) and a sequence of contextualized embeddings \( o_{ni} \in \mathbb{R}^d \), one for each token. For each model, we use an Adam optimizer with \( \eta = 0.001, \beta_1 = 0.9, \beta_2 = 0.999 \), and \( \epsilon = 10^{-7} \) to minimize the empirical loss across samples in each batch.

4 METHODS

4.1 Number entity recognition

BERT, introduced by [3], has been shown to exhibit state-of-the-art performance for NER. Token embeddings are passed through a Transformer encoder, generating contextualized representations for each subword token, which are classified by the attention-based BERT classifier. We fine-tune the BERT model on the SQuAD-Num dataset. Both questions and context for each sequence are input to the NuER model, and the model outputs a sequence of tags denoting whether each token is one of the six types of numbers or "other" (seven total classes). The outputs of the NuER model for each element in the sequence, given by the \( o_i \), are mapped through a linear layer (we drop the \( n \) subscript for clarity), and the resultant scalars are fed into a softmax function:

\[
y^{1}_{i} = \frac{\exp(W_o o_i)}{\sum_{k=1}^{K} \exp(W_o o_k)}
\]

where \( W \in \mathbb{R}^{d}, W = [w_1, \ldots, w_k, \ldots, w_K]^T \in \mathbb{R}^{K \times d} \) are the parameters of the linear layer, \( k = 1, \ldots, K \) indexes the entity labels, and the exp operator is applied elementwise. The model is trained with categorical cross-entropy loss:

\[
L = \sum_{i=1}^{T} \sum_{k=1}^{K} -y_{ik} \log(y^{1}_{ik})
\]

where \( y_{ik} \in \{0, 1\} \) denotes whether the \( i \)th element belongs to class \( k \). We divide SQuAD-Num into 75% training, 10% validation, and 15% testing. We fine-tune the NuER model for 20 epochs on SQuAD-Num and evaluate our model on the out-of-domain HA-Numeracy 2K.

4.2 Jointly trained embeddings

We experiment with training embeddings for tokens jointly with number entities, referred as Jointly trained Embeddings (JEM) [18]. Specifically, we compare the performance of numerical question...
an answering on SQuAD-Num, with and without entity embeddings. As input to the pretrained model, we feed in pairs \((v_i, z_i)\) of tokens \(v_i\) and number entity labels \(z_i, z_i \in \{0, 1\}^{K+1}\) is a one-hot vector, where \(K = 6\) indicates the six entities of numbers in the corpus. During embedding lookup, the pairs of embeddings for each token are summed and fed as inputs:

\[
\hat{e}_i = e_i + h_i
\]

where \(e_i, h_i\) is the trainable embedding and number entity embedding, respectively for token \(i\). We compare the performance of this model after fine-tuning for two epochs against the baseline, which is only fed the token identities as input. As is standard for SQuAD, the model is trained with cross-entropy loss in order to predict the probability of any token in the context being the start/end of a span that answers the numerical question:

\[
\mathcal{L} = \sum_{i=1}^{T} -y_i \log(\hat{y}_i)
\]

where here, \(y_i \in \{0, 1\}\) identifies the start/end of a span and \(\hat{y}_i\) is the corresponding probability predicted by our model.

### 4.3 FITB

In this subsection, we measure the effectiveness of NuER on classification tasks. We handcraft a Fill-In-The-Blank (FITB) task of predicting a masked number in a given input sentence. This is challenging as there are several thousand unique numbers in the corpus to which a masked number could be classified. It measures the true capability of BERT to fill in a number given the context. Our model which we refer to as \(\text{BERT}_{\text{NuER}}\) consists of BERT tokenizer with six special tokens namely \(<\text{size}>, <\text{age}>, <\text{count}>, <\text{date}>, <\text{year}>,\) and \(<\text{percentage}>\). These special tokens are added next to the CLS as an indication to the model about their respective number entity types. On the classification side, a linear projection layer of size \(M \times D\) is created to learn about the six different entity types \((M = 6, D = 768)\). These projections are then concatenated with the BERT hidden states of 768 dimensions to obtain final set of dimensions \((768 \times 2) \times M\) which is projected again to a classification layer of \(V\) output vocabulary items. Here, the \(V\) output vocabulary items represent the unique numbers that are found in the corpus.

### 5 RESULTS AND DISCUSSION

#### 5.1 Number entity recognition

Table 2 shows the performance of our NuER model on each dataset. We find that the model performs well across all categories on SQuAD-Num and in most categories of ENT-Numeracy 600K. Performance drops are observed for the "Age" and "Size" categories, partially because there are not enough training instances of these two categories in SQuAD-Num. For "Age," precision is high while recall is low. This is a limitation of our work and could be because the numbers that are classified as "Age" can typically be confused with other entities including "Size," suggesting that the model is only able to capture age accurately when it is quite obvious in the dataset. For "Size," both precision and recall are low, perhaps because, unlike the other types of numbers, size may have many different units (e.g., feet, meters, acres, etc.), which make it more difficult to associate with the numerical value. Table 3 shows that the classification performance when only few of the samples are used in fine-tuning is still comparable to the full fine-tuning performance on two classes. Though it works well for few classes, the F1 scores drop significantly for classes like Age and Size.

#### 5.1.1 Training settings

For Table 2 experiments, the learning rate was set to \(2 \times 10^{-5}\). For few-shot used in Table 3, only first 20 samples (using grid search, fewer samples to retain maximum accuracy compared to full finetuning) from Year and Count, while full finetuning uses all the samples.

#### 5.2 Jointly trained embeddings

Table 4 compares the performance of our model trained on numerical question answering against the BERT baseline. Our model demonstrates gains in both the exact match and F1 scores, suggesting that the additional information imparted by knowledge of the type of numbers that are present in the question and context aids the model in localizing the number within the context. A variety of models including \(\text{BERT}_q\) (base), \(\text{BERT}_l\) (large) and \(\text{RoBERTa}_b\) (base) \([3, 5]\) are used to understand if these improvements are consistent. We found that in terms of overall performance, \(\text{RoBERTa}_b > \text{BERT}_l >\text{BERT}_q\), with \(\text{RoBERTa}_b\) being the highest, while in terms

### Table 2: Performance metrics for NuER on SQuAD-Num and ENT-Numeracy 600K for each of the six entity types.

|                | Year | Count | Percentage | Age | Size | Date | Total     |
|----------------|------|-------|------------|-----|------|------|-----------|
| SQuAD-Num      |      |       |            |     |      |      |           |
| Precision      | 98.61| 88.66 | 95.08      | 94.12| 87.84| 100.00| 95.38     |
| Recall         | 98.99| 95.91 | 96.67      | 88.89| 84.42| 100.00| 97.24     |
| F1             | 98.8 | 92.15 | 95.87      | 91.43| 86.09| 100.00| 96.3      |
| ENT-Numeracy 600K |      |       |            |     |      |      |           |
| Precision      | 97.96| 92.00 | 91.67      | 75.00| 54.55| 96.88| 94.26     |
| Recall         | 96.48| 81.01 | 95.65      | 7.14 | 30.00| 85.94| 84.14     |
| F1             | 97.21| 86.16 | 93.62      | 13.04| 38.71| 91.09| 88.91     |

### Table 3: Few-shot transfer vs fine-tuning performance on NuER entity types.

|               | Year | Count |
|---------------|------|-------|
| Few-shot      |      |       |
| Precision     | 87.09| 76.27 |
| Recall        | 97.32| 89.59 |
| F1            | 91.92| 82.39 |
| Full finetuning |    |       |
| Precision     | 91.15| 80.53 |
| Recall        | 98.49| 95.33 |
| F1            | 94.68| 87.31 |
of performance improvements, $BERT_L > RoBERTa_B > BERT_B$ with $BERT_L$ JEM obtaining an improvement of 1.64 exact match and 1.7 F1 points respectively.

5.2.1 Training settings. BERT for QA consists of a pretrained model along with classification heads for start and ends of spans. The model was trained for 2 epochs with a learning rate of $2 \times 10^{-5}$. During training, embeddings for numbers and their entities are learned in parallel.

5.3 FITB

Table 5 shows the performance improvements of $BERT_{NuER}$ over the baseline for FITB. The top-$k$ performance tells us that $BERT_{NuER}$ is able to predict the correct answer within the top-$k$ entries. We see that the gap in performance improvement increases as $k$ increases. This tells us that the enriched model is more accurate in its predictions. The dist metric quantifies the average absolute distance between pairs of actual value and the predicted values. This ensures the quality of numerical predictions. For example, as shown in the first column of Figure 2, when the ground truth is a Year, we expect all the other predictions to also be years (as opposed to Count or other classes). The dist metric calculates the spread of numbers while top-$k$ calculates the accuracy itself. We see that the dist is significantly lower than the baseline model. Figure 2 shows that $BERT_{NuER}$ generates top-$k$ predictions whose elements are more consistent with the desired class label.

5.3.1 Training settings. For the FITB experiment, DistilBERT [9] with an Adam optimizer and a learning rate of $2 \times 10^{-4}$ is used. The model is trained for 3 epochs with a batch size of 64. On the architectural side, adding a linear layer to the BERT hidden states yielded marginal improvements. But, concatenating the linear layer responsible for learning number entity onto the BERT hidden states followed by classification lead to significant improvements.

6 CONCLUSIONS

We have defined number entities and find that this aids multiple NLP tasks, via joint training, and classification. We also introduce a dataset representing number entities and a methodology to annotate unseen numbers. In the classification objective, we find that the addition of entities provides a considerable boost in FITB evident through the quality of predictions. Through our datasets and methods, we hope to incite further research at the intersection of numeracy and NER.
REFERENCES

[1] Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving bert a calculator: Finding operations and arguments with reading comprehension. arXiv preprint arXiv:1909.00109 (2019).

[2] Chung-Chi Chen, Hsin-Hsien 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. 6307–6313.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).

[4] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360 (2016).

[5] 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 (2019).

[6] 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. 3374–3380.

[7] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250 (2016).

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems. 5998–6008.

[9] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. arXiv preprint arXiv:1805.04174 (2018).