Methods for Numeracy-Preserving Word Embeddings

Dhanasekar Sundararaman[1], **Shijing Si**[1], **Vivek Subramanian**[1],
Guoyin Wang[2], **Devamanyu Hazarika**[3], **Lawrence Carin**[1]
[1] Duke University
[2] Amazon Alexa AI
[3] National University of Singapore
dhanasekar.sundararaman@duke.edu

Abstract

Word embedding models are typically able to capture the semantics of words via the distributional hypothesis, but fail to capture the numerical properties of numbers that appear in a text. This leads to problems with numerical reasoning involving tasks such as question answering. We propose a new methodology to assign and learn embeddings for numbers. Our approach creates Deterministic, Independent-of-Corpus Embeddings (referred to as DICE) for numbers, such that their cosine similarity reflects the actual distance on the number line. DICE outperforms a wide range of pre-trained word embedding models across multiple examples of two tasks: (i) evaluating the ability to capture numeration and magnitude; and (ii) to perform list maximum, decoding, and addition. We further explore the utility of these embeddings in downstream applications by initializing numbers with our approach for the task of magnitude prediction. We also introduce a regularization approach to learn model-based embeddings of numbers in a contextual setting.

1 Introduction

Word embeddings capture semantic relationships between words by operationalizing the distributional hypothesis (Harris, 1954; Firth, 1957). They can be learned either non-contextually (Mikolov et al., 2013b; Pennington et al., 2014; Bojanowski et al., 2017) or contextually (Devlin et al., 2018; Peters et al., 2018). Non-contextual embeddings have worked well on various language understanding and semantic tasks (Rumelhart et al., 1988; Mikolov et al., 2013a,b). More recently, they have also been used as pre-trained word embeddings to aid more sophisticated contextual models for solving rigorous natural language processing (NLP) problems, including translation, paraphrasing, and sentence-similarity tasks (Kiros et al., 2015; Wieting et al., 2015).

While word embeddings effectively capture semantic relationships between *words*, they are less effective at capturing numeric properties associated with *numbers*. Though numbers represent a significant percentage of tokens in a corpus, they are often overlooked. In non-contextual word embedding models, they are treated like any other word, which leads to misinterpretation. For instance, they exhibit unintuitive similarities with other words and do not contain strong prior information about the magnitude of the number they encode. In sentence similarity and reasoning tasks, failure to handle numbers causes as much as 29% of contradictions (De Marneffe et al., 2008). In other data-intensive tasks where numbers are abundant, like neural machine translation, they are masked to hide the translation models inefficiency in dealing with them (Mitchell and Lapata, 2009).

There are a variety of tests proposed to measure the efficiency of number embeddings. For instance, Naik et al. (2019) shows that GloVe (Pennington et al., 2014), word2vec (Mikolov et al., 2013b), and fastText (Joulin et al., 2016; Bojanowski et al., 2017) fail to capture *numeration* and *magnitude* properties of a number. Numeration is the property of associating numbers with their corresponding word representations ("3" and "three") while magnitude represents a number's actual value ($3 < 4$). Further, Wallace et al. (2019) proposes several tests for analyzing numerical reasoning of number embeddings that include list maximum, decoding, and addition.

In this paper, we experimentally demonstrate that if the cosine similarity between word embeddings of two numbers reflects their actual distance on the number line, the resultant word embeddings are useful in downstream tasks. We first demonstrate how *Deterministic, Independent-of-Corpus Embeddings* (DICE) can be constructed such that they almost perfectly capture properties of numera-

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 4742–4753,
November 16–20, 2020.

tion and magnitude. These non-contextual embeddings also perform well on related tests for numeracy (Wallace et al., 2019).

To demonstrate the efficacy of DICE for downstream tasks, we explore its utility in two experiments. First, we design a DICE embedding initialized Bi-LSTM network to classify the magnitude of masked numbers in the 600K dataset (Chen et al., 2019). Second, given the popularity of modern contextual model-based embeddings, we devise a regularization procedure that emulates the hypothesis proposed by DICE and can be employed in any task-based fine-tuning process. We demonstrate that adding such regularization helps the model internalize notions of numeracy while learning task-based contextual embeddings for the numbers present in the text. We find promising results in a numerical reasoning task that involves numerical question answering based on a sub-split of the popular SQuAD dataset (Rajpurkar et al., 2016).

Our contribution can be summarized as follows:

- We propose a deterministic technique to learn numerical embeddings. DICE embeddings are learned independently of corpus and effectively capture properties of numeracy.
- We prove experimentally that the resultant embeddings learned using the above methods improve a model's ability to reason about numbers in a variety of tasks, including numeration, magnitude, list maximum, decoding, and addition.
- We also demonstrate that properties of DICE can be adapted to contextual models, like BERT (Devlin et al., 2018), through a novel regularization technique for solving tasks involving numerical reasoning.

2 Related Work

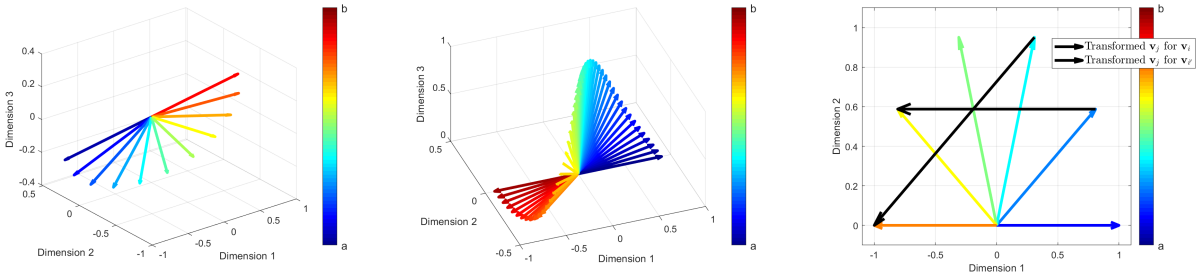
The major research lines in this area have been dedicated to (i) devising probing tests and curating resources to evaluate the numerical reasoning abilities of pre-trained embeddings, and (ii) proposing new models that learn these properties.

Naik et al. (2019) surveyed a number of non-contextual word embedding models and highlighted the failure of those models in capturing two essential properties of numbers – *numeration* and *magnitude*. Chen et al. (2019) created a novel dataset named Numeracy-600k, a collection of approximately 600,000 sentences from market comments with a diverse set of numbers representing age, height, weight, year, etc. The authors use neural network models, including a GRU, BiGRU, CRNN, CNN-capsule, GRU-capsule, and BiGRU-capsule, to classify the magnitude of each number. Wallace et al. (2019) compares and contrasts the numerical reasoning ability of a variety of non-contextual as well as contextual embedding models. The authors also proposed three tests – list maximum, decoding, and addition – to judge the numerical reasoning ability of embeddings of numerals. They infer that word embedding models that perform the best on these three tests have captured the numerical properties of numbers well. Therefore, we consider these proposed tests in our evaluation. (Spithourakis and Riedel, 2018) used a variety of models to distinguish numbers from words, and demonstrated that this ability reduces model perplexity with neural machine translation. Weiss et al. (2018) found that neural networks are capable of reasoning numbers with explicit supervision.

Numerically Augmented QANet (NAQANet) (Dua et al., 2019) was built by adding an output layer on top of QANet (Yu et al., 2018) to predict answers based on addition and subtraction over numbers in the DROP dataset. Our work, in contrast, offers a simple methodology that can be added to any model as a regularization technique. Our work is more similar to Jiang et al. (2019), where embedding of a number is learned as a simple weighted average of its prototype embeddings. Such embeddings are used in tasks like word similarity, sequence labeling and have been proven to be effective.

3 Methods

To overcome NLP models inefficiency in dealing with numbers, we consider our method DICE to form embeddings. To begin, we embed numerals and word forms of numbers as vectors $\mathbf{e}_i \in \mathbb{R}^D$, where i indexes numerals identified within a corpus. We first preprocess by parsing the corpora associated with each of our tasks (described below) for numbers in numeral and word forms to populate a number vocabulary. Then, the dimensionality of the embeddings required for that task is fixed. We explicitly associate the embeddings of a numeral and word forms of numbers to have the same embedding.



(a) DICE-2 embedding (b) DICE-D embedding (c) Addition of DICE vectors

Figure 1: Proposed DICE embeddings. Vectors are colored according to numeral magnitude. Note that addition of two numbers in this embedding is performed by a shift, scaling, and rotation. Scaling depends only on the vector being added, as illustrated in sub-figure (c) in which the two black lines, corresponding to identical $\mathbf{e}_j$, have the same length.

3.1 DICE embeddings

In designing embeddings that capture the aforementioned properties of numeration and magnitude, we consider a deterministic, handcrafted approach (depicted in Figures 1a and 1b). This method relies on the fact that tests for both numeration and magnitude are concerned with the correspondence in similarity between numbers in token space and numbers in embedding space. In token space, two numbers $x, y \in \mathbb{R}$, in numeral or word form (with the latter being mapped to its corresponding numeral form for comparison), can be compared using absolute difference, $i.e.$:

$$d_n(x, y) = |x - y| \tag{1}$$

The absolute value ensures that two numbers are treated as equally distant regardless of whether $x \geq y$ or $y \geq x$. On the other hand, two embeddings $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$ are typically compared via cosine similarity, given by:

$$s_e(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T \mathbf{y}}{||\mathbf{x}||_2 ||\mathbf{y}||_2} = \cos(\theta) \tag{2}$$

$$d_e(\mathbf{x}, \mathbf{y}) = 1 - \cos(\theta) \tag{3}$$

where θ is the angle between $\mathbf{x}$ and $\mathbf{y}$ and $d_e(\mathbf{x}, \mathbf{y})$ is their cosine distance. Normalization by the vector lengths ensures that the metric is independent of the lengths of the two vectors.

Note that numerals are compared in terms of distance while their embeddings are compared by similarity. As cosine distance increases, the angle between $\mathbf{x}$ and $\mathbf{y}$ increases monotonically. A distance of zero is achieved when $\mathbf{x}$ and $\mathbf{y}$ are oriented in the same direction. When $\mathbf{x} \perp \mathbf{y}$, the cosine distance is 1; and when $\mathbf{x}$ and $\mathbf{y}$ are antiparallel, cosine distance is 2.

We seek a mapping $(x, y) \mapsto (\mathbf{x}, \mathbf{y})$ such that d_e monotonically increases as d_n increases. We first bound the range of numbers for which we wish to compute embeddings by $[a, b] \subset \mathbb{R}$ and, without loss of generality, restrict $\mathbf{x}$ and $\mathbf{y}$ to be of unit length ($i.e.$, $||\mathbf{x}||_2 = ||\mathbf{y}||_2 = 1$). Since the cosine function decreases monotonically between 0 and π, we can simply employ a linear mapping to map distances $s_n \in [0, |a - b|]$ to angles $\theta \in [0, \pi]$:

$$\theta(s_n) = \frac{s_n}{|a - b|}\pi \tag{4}$$

This mapping achieves the desired direct relationship between s_n and d_e. Since there are infinitely many choices for $\mathbf{x}$ and $\mathbf{y}$ with angle θ, we simply fix the direction of the vector corresponding to the numeral a. Numbers that fall outside $[a, b]$ are mapped to a random angle in $[-\pi, \pi]$. In the corpora we considered, a and b are chosen such that numbers outside $[a, b]$ represent a small fraction of the total set of numbers (approximately 2%).

We employ this mapping to generate numeral embeddings in $\mathbb{R}^D$. Figure 1a shows deterministic, independent-of-corpus embeddings of rank 2 (DICE-2). In this approach we represent angles as vectors in $\mathbb{R}^2$ using the polar-to-Cartesian coordinate transformation:

$$[r, \theta] \mapsto [x_1, x_2] = [r\cos(\theta), r\sin(\theta)]\mathbf{v} \tag{5}$$

where we choose $r = 1$ without loss of generality. We then sample a random matrix $\mathbf{M} \in \mathbb{R}^{D \times D}$ where $D \geq 2$ and $m_{ij} \sim \mathcal{N}(0, 1)$ and perform a QR decomposition on $\mathbf{M}$ to obtain a matrix $\mathbf{Q}$

Model	OVA	SC	BC
Random	0.04	48.92	49.34
Glove 6B-200D	15.88	62.21	83.94
Glove 6B-300D	18.41	62.92	83.98
Glove-840B-300D	5.18	55.58	91.86
FastText-Wiki	13.94	59.96	96.15
FastText-CC	7.83	53.89	85.40
Skip-gram-5	8.85	55.40	96.42
Skip-gram-Dep	3.32	51.99	94.60
DICE-D (ours)	**95.63**	**99.66**	**99.64**

Table 1: Performance (% accuracy) on numeracy tests.

whose columns $\mathbf{q}_i, i = 1, \ldots, D$ constitute an orthonormal basis for $\mathbb{R}^D$. The DICE-2 embedding $\mathbf{e} \in \mathbb{R}^D$ of each numeral is then given by $\mathbf{e} = \mathbf{Q}_{1:2}\mathbf{v}$, where the subscript on $\mathbf{Q}$ indicates taking the first two columns of $\mathbf{Q}$.

In Figure 1b we consider DICE-D, in which we generate vectors in $\mathbb{R}^D$ by applying a polar-to-Cartesian transformation in D dimensions (Blumenson, 1960):

$$v_d = \begin{cases} [\sin(\theta)]^{d-1}\cos(\theta), & 1 \leq d < D \\ [\sin(\theta)]^{D}, & d = D \end{cases} \quad (6)$$

where the subscripts indicate the coordinate in $\mathbf{v}$. We again apply a QR-decomposition on a random matrix $\mathbf{M}$ generated as above, except here we project $\mathbf{v}$ using all D basis vectors. This allows for a random rotation of the embeddings to avoid bias due to choosing $e_{a1} = 1, e_{ai} = 0 \ \forall i \neq 1$. We employ DICE-D embeddings throughout this paper as word embeddings are practically not 2 dimensional.

4 Experiments

To observe the numerical properties of DICE, we consider two tasks: **Task 1** deals with the *numeration (NUM)* and *magnitude (MAG)* properties as proposed by (Naik et al., 2019); **Task 2** performs *list maximum*, *decoding*, and *addition* as proposed by (Wallace et al., 2019). We then experiment on two additional tasks to demonstrate the applications of DICE.

4.1 Task 1: Exploring Numeracy

In this task, proposed by Naik et al. (2019), there are three tests for examining each property of numeration (NUM, 3 = "three") and magnitude (MAG, $3 < 4$). For each of these tests, target numbers in its word or numeral form are evaluated against other numbers as follows:

- *One-vs-All (OVA):* The distance between the embedding vector of the target and its nearest neighbor should be smaller than the distance between the target and any other numeral in the data.
- *Strict Contrastive (SC):* The distance of the embedding vector of the target from its nearest neighbor should be smaller than its second nearest neighbor numeral.
- *Broad Contrastive (BC):* The distance of the embedding vector of the target numeral from its nearest neighbor should be smaller than its furthest neighbor.

Training Details. We use the Gigaword corpus obtained from the Linguistic Data Consortium to populate the list of numbers from the dataset. Parsing was performed using the text2digits[1] Python module. As done by Naik et al. (2019), we employ $D = 300$ for the DICE-D embeddings. Embeddings of numerals are assigned using the principle explained in Section 3.1, while the embedding of words that denote numbers (word form) simply points to the embedding of that numeral itself. We then perform the six tests (OVA-NUM / OVA-MAG, SC-NUM/ SC-MAG, BC-NUM / BC-MAG) on 130 combinations of numbers for NUM and $31,860$ combinations of numbers for MAG.

Evaluation. Following Naik et al. (2019), we use accuracy to measure the efficiency of the embeddings. These tests require the fulfillment of certain clauses which are defined in Naik et al. (2019).

Results. Table 1 shows comparisons of the performance of embeddings created by each of the DICE methods on the MAG tests. Compared to the baselines, both DICE methods outperform all commonly employed non-contextual word embedding models in OVA, SC, and BC tests. This is attributed to the cosine distance property addressed in the DICE embeddings. Specifically, because the magnitude of the number is linearly related to its angle, sweeping through numbers in order guarantees an increase in angle along each axis. Numbers that are close to each other in magnitude are rotated further but in proportion to their magnitude. Thus, small and large numbers are ensured to lie

[1] https://pypi.org/project/text2digits/

	List maximum (accuracy)			Decoding (RMSE)			Addition (RMSE)		
Integer range	[0, 99]	[0, 999]	[0, 9999]	[0, 99]	[0, 999]	[0, 9999]	[0, 99]	[0, 999]	[0, 9999]
Random vectors	0.16	0.23	0.21	29.86	292.88	2882.62	42.03	410.33	4389.39
Untrained CNN	0.97	0.87	0.84	2.64	9.67	44.40	1.41	14.43	69.14
Untrained LSTM	0.70	0.66	0.55	7.61	46.5	210.34	5.11	45.69	510.19
Value embedding	**0.99**	0.88	0.68	1.20	11.23	275.50	**0.30**	15.98	654.33
Pretrained									
Word2Vec	0.90	0.78	0.71	2.34	18.77	333.47	0.75	21.23	210.07
GloVE	0.90	0.78	0.72	2.23	13.77	174.21	0.80	16.51	180.31
ELMo	0.98	0.88	0.76	2.35	13.48	62.20	0.94	15.50	45.71
BERT	0.95	0.62	0.52	3.21	29.00	431.78	4.56	67.81	454.78
Learned									
Char-CNN	0.97	**0.93**	0.88	2.50	4.92	11.57	1.19	7.75	**15.09**
Char-LSTM	0.98	0.92	0.76	2.55	8.65	18.33	1.21	15.11	25.37
DROP-trained									
NAQANet	0.91	0.81	0.72	2.99	14.19	62.17	1.11	11.33	90.01
NAQANet (w/out GloVe)	0.88	0.90	0.82	2.87	5.34	35.39	1.45	9.91	60.70
Ours									
DICE-D	0.98	0.87	**0.96**	**0.43**	**0.83**	**3.16**	0.75	**2.79**	29.95

Table 2: Experimental results on list maximum, decoding, and addition using the DICE-D method.

near other small and large numbers, respectively, in terms of cosine distance.

On the NUM tests, DICE achieves perfect accuracy. The primary reason DICE embeddings perform so well on numeracy tasks is that the preprocessing steps taken allow us to parse a corpus for word forms of numbers and explicitly set matching embeddings for both word and numeral forms of numbers. Each of these embeddings is guaranteed to be unique since a number's embedding is based on its magnitude, $i.e.$, the larger the magnitude, the greater the angle of the embedding, with a maximum angle of π. This ensures that the numeral form of a number is always able to correctly identify its word form among all word forms in the corpus as that with the smallest cosine distance (which equals zero). Performance on OVA-NUM is a lower bound on the performance of SC-NUM and BC-NUM, so those tests are guaranteed to pass under our approach.

4.2 Task 2: List Maximum, Decoding, and Addition

This task considers the operations proposed by (Wallace et al., 2019) – list maximum, decoding, and addition. List maximum deals with the task of predicting the maximum number given the embedding of five different numbers. Decoding deals with regressing the value of a number given its embedding. An additional task involves predicting the sum of two numbers given their embeddings.

Training Details. The list-maximum test presents to a Bi-LSTM neural network a set of five numbers of the same magnitude, and the network is trained to report the index of the maximum number. In the decoding test, a linear model and a feed-forward network are each trained to output the numeral corresponding to the word form of a number based on its embedding. Finally, in the addition test, a feed-forward network is trained to take in the embeddings of two numbers as its input and report the sum of the two numbers as its output. Each test is performed on three ranges of integers $[0, 99]$, $[0, 999]$, and $[0, 9999]$, with an 80/20 split of training and testing data sampled randomly. The neural network is fed with the embedding of numbers; the task is either classification (in the case of list maximum) or prediction of a continuous number (in case of addition and decoding). We replicate the exact experimental conditions and perform the three tests with DICE embeddings. For the sake of consistency with the tests proposed by (Wallace et al., 2019), we also only deal with positive in this experiment.

Evaluation. List maximum again uses accuracy as its metric while decoding and addition use root mean squared error (RMSE), since predictions are continuous.

Results. Given the strong performance of the DICE-D method on the NUM and MAG tests, we next consider its performance on tasks involv-

ing neural network models. In their empirical study, (Wallace et al., 2019) compared a wide range of models that included a random baseline; character level models such as a character-CNN and character-LSTM, which were both untrained and trained; a so-called value embedding model in which numbers are embedded as their scalar value; traditional non-contextual word embedding models including Word2Vec and GloVe; contextual word embedding models including ELMo and BERT; and the Numerically Aware Question Answering (NAQA) Network, a strong numerical reasoning model proposed on the Discrete Reasoning over Paragraphs (DROP) dataset.

We compare the performance of our DICE-D embedding to that of the other models on each of the three tasks proposed by (Wallace et al., 2019). Results are presented in Table 2. We find that our DICE embedding exceeds the performance of more sophisticated models by large margins in all but four cases. In two of those four, our model fell short by only a few percentage points. We attribute the success of the DICE-D approach to the fact that the model is, by design, engineered to handle numeracy. Just as the value embedding model – which proved to be reasonably successful in all three tasks across a wide range of numbers – captures numeracy through the magnitude of embeddings, our model captures numeracy through the angle corresponding to the embeddings.

The value embedding model, however, breaks down as the range of the processed numbers grows. This is likely because, as demonstrated by Trask et al. (2018), networks trained on numeracy tasks typically struggle to learn an identity mapping. We reason that our model outperforms the value embedding model because the network learns to associate features between the set of inputs such that the input vectors can be scaled, rotated, and translated in D dimensions to achieve the desired goal.

More precisely, for a neural network to learn addition, numbers must be embedded such that their vector embeddings can be consistently shifted, rotated, and scaled to yield the embedding of another number (see Figure 1c). The choice of embedding is essential as it may be impractical for a network to learn a transformation for all embeddings that obeys this property (without memorization).

DICE is quite similar to the value embedding system, which directly encodes a number's value in its embeddings. However, DICE performs better due to its compatibility with neural networks, whose layers are better suited for learning rotations and scaling than identity mappings.

Finally, both the value embedding models for a small number range and the character level models remain somewhat competitive, suggesting again that exploring a digit-by-digit embedding of numerals may provide a means of improving our model further.

5 Applications of DICE

5.1 Magnitude Classification

We examine the importance of good initialization for number embedding vectors (Kocmi and Bojar, 2017), particularly for better contextual understanding. In particular, we experiment on the magnitude classification task, which requires the prediction of magnitudes for masked numbers. The task is based on the 600K dataset proposed by Chen et al. (2019), which requires classification into one of seven categories corresponding to powers of 10 in $\{0, 1, 2, 3, 4, 5, 6\}$.

Training Details. We use a bi-LSTM (Hochreiter and Schmidhuber, 1997) with soft attention (Chorowski et al., 2015) to classify the magnitude of masked numbers. Numerals are initialized with corresponding DICE embeddings, and the target number is masked by substituting a random vector. Each token x_n in a sequence of length N is associated with a forward and backward LSTM cell. The hidden state $\mathbf{h_n}$ of each token is given by the sum of the hidden states of the forward and backward cells: $\mathbf{h}_n = \overleftarrow{\mathbf{h}}_n + \overrightarrow{\mathbf{h}}_n$. To generate a context vector $\mathbf{c}$ for the entire sentence, we compute attention scores α_n by taking the inner product of each hidden state $\mathbf{h}_n$ with a learned weight vector $\mathbf{w}$. The resulting scores are passed through a softmax function, and the weights are used to form a convex combination of the $\mathbf{h}_n$ that represents the context $\mathbf{c}$ of the sentence. Logits are obtained by taking the inner product of $\mathbf{c}$ with trained embeddings for each of the seven categories, and cross-entropy loss is minimized. More details on training can be found in Appendix A.

Evaluation. Following Chen et al. (2019), we use micro and macro F1 scores for classifying the magnitude of a number.

Results. Table 3 shows significant improvements in the F1 score achieved by the model. To investigate the effects of dimensions of the embedding

Model	Micro-F1	Macro-F1
LR	62.49	30.81
CNN	69.27	35.96
GRU	70.92	38.43
BiGRU	71.49	39.94
CRNN	69.50	36.15
CNN-capsule	63.11	29.41
GRU-capsule	70.73	33.57
BiGRU-capsule	71.49	34.18
BiLSTM with DICE	**75.56**	**46.80**

Table 3: Performance (%) on classifying number magnitude on the Numeracy-600k dataset.

Embedding Size	Micro-F1	Macro-F1
32	74.63	45.92
64	74.90	45.99
128	75.55	46.36
256	**75.56**	45.56
512	74.14	**46.80**

Table 4: Performance (%) of BiLSTM-attention with DICE model on the Numeracy-600k dataset by varying the embedding dimensions of input tokens.

and hidden vectors within the LSTM cells on the performance of the BiLSTM-attention model, we perform ablation experiments. We vary the embedding size of tokens while keeping other hyperparameters constant, and observe the results on Tables 4. From Table 4 the BiLSTM with DICE model achieves the best micro-F1 score when the embedding dimension is 256. However, the macro-F1 score peaks when the embedding dimension is 512.

These results suggest that while DICE embeddings yield superior performance in non-contextual numerical tasks, such as computing the maximum and performing basic mathematical operations, data agnostic embeddings such as DICE may not be ideal for textual reasoning tasks in which words surrounding a number provide important information regarding the magnitude of the number. Hence, we introduce a model-based regularization method that utilizes the DICE principles to learn number embeddings in 5.2.

5.2 Model-Based Numeracy Embeddings

In the previous section, we demonstrated how DICE could be explicitly incorporated for numbers in the text. Here, we propose a methodology that help models implicitly internalize the properties of DICE. Our approach involves a regularization method (an auxiliary loss) that can be adopted in the fine-tuning of any contextual NLP model, such as BERT. Auxiliary losses have shown to work well for a variety of NLP downstream tasks (Shen et al., 2019).

During the task-specific training of any model, the proposed auxiliary loss $\mathcal{L}_{num}$ can be applied to the input embeddings of numbers available in a minibatch. For any two contextual numerical embeddings $\mathbf{x}, \mathbf{y}$ obtained from the final hidden layer of the model, the $\mathcal{L}_{num}$ loss for the pair of numbers (x, y) is calculated as:

$$\mathcal{L}_{num} = \left\| 2\frac{|x-y|}{|x|+|y|} - d_{\cos}(\mathbf{x},\mathbf{y}) \right\|_2 \quad (7)$$

where $d_{\cos}(\mathbf{x},\mathbf{y}) = 1 - \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2}$ is the cosine distance between the embeddings $\mathbf{x}$ and $\mathbf{y}$. In essence, $\mathcal{L}_{num}$ follows the same motivation as DICE where cosine distance between the embeddings of two numbers are encouraged to be proportional to their (scaled) absolute magnitude distance on the number line.

Training Details. To evaluate the proposed $\mathcal{L}_{num}$, we test the regularization on the task of question answering (QA) involving numerical answers. In particular, we take the popular Stanford Question Answering Dataset (SQuAD 1.1) (Rajpurkar et al., 2016) dataset and create sub-splits (ranges from [1, 30000]) where the (i) training QA pairs have answers strictly containing numerical digits (Sub-split 1, less than 10K examples), and (ii) training QA pairs have answers containing a number as one of their tokens, for e.g. "10 apples" (Sub-split 2, slightly more than 10K examples). We create these splits to evaluate BERT model's reasoning involving numbers to pick these answers. We choose `BERT-base-uncased` as baseline model and train it on both the datasets. Within each batch, we calculate $\mathcal{L}_{num}$ by randomly sampling a pair of numbers x, y from the available numbers in the contexts. The corresponding embeddings of the numbers are $\mathbf{x}$ and $\mathbf{y}$, which are extracted from the last hidden layer of the BERT model. We then enforce the distance of embeddings to match the difference between number values by $\mathcal{L}_{num}$. The scores are reported on the development set (less than 1000 examples) as the test set cannot be pruned for our purpose. The assumption here is that the BERT model needs to perform numerical reasoning to come up with answers for these particular kinds of QA pairs. The models were trained on Nvidia Tesla P100 GPU. More details on choosing

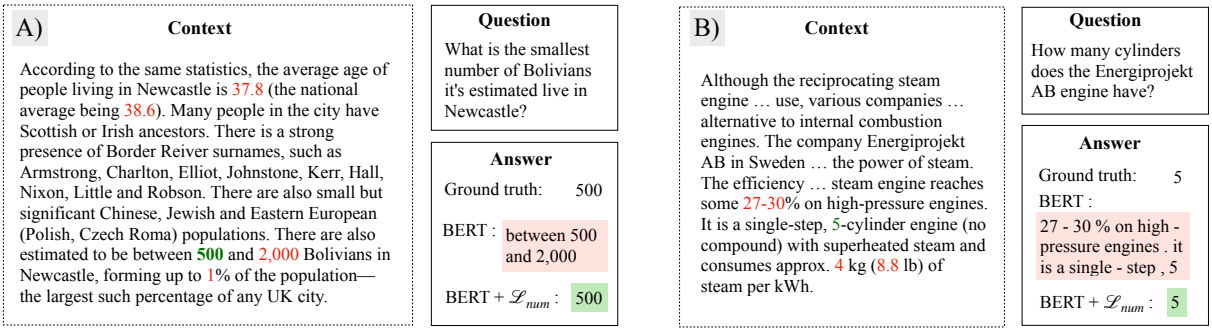


Figure 2: Qualitative examples where BERT + $\mathcal{L}_{num}$ performed better than BERT-base

hyper-parameter for BERT + $\mathcal{L}_{num}$ is discussed in Appendix B.

Evaluation. Exact Match is a binary measure ($i.e.$, true/false) of whether the predicted output matches the ground truth answer exactly. This evaluation is performed after the string normalization (uncased, articles removed, etc.). F1 is the harmonic mean of precision and recall.

Results. Results in Table 5 show that the BERT model with numeracy objective achieves an improvement of 0.48 F1 points when the answers are purely numerical digits. When the BERT model is trained on QA pairs with answers containing at least a number with several words, and evaluated on pairs with answers containing only numbers, we see an improvement of 1.12 F1 points over the baseline model.

The BERT-base model on the original SQuAD data was finetuned for 3 epochs owing to its complexity. However, we find that 1 epoch is sufficient to capture the complexity of the pruned SQuAD data. Table 5 shows BERT + $\mathcal{L}_{num}$ consistently performs better than BERT-base across epochs.

Interestingly, BERT-base performs worse when finetuned with QA pairs containing a mix of words and numbers as answers (sub-split 2). This informs us that the baseline model learns to pick numbers better but fails to do as well when fine-tuned with a mix of words and numbers. In both the cases, the evaluation set consists of pruned SQuAD dev set QA pairs with answers strictly containing numerical digits only. We find that BERT + $\mathcal{L}_{num}$ gives the maximum improvement on sub-split 2 data highlighting the efficiency of our regularization technique to learn numerical embeddings.

Figure 2 shows some qualitative examples where the BERT + $\mathcal{L}_{num}$ performs better than BERT-base (Sub-split 2). In this analysis, we found that the baseline model picks the whole sentence or paragraph involving the numerical value (Figure 2 B) as the answer. Our method picks numbers within the classification span (Figure 2 B) and sometimes helps the BERT model to accurately pick up correct numbers (Figure 2 A), contributing to exact match and F1. More such examples are shown in Appendix C.

Model	Epochs	Sub-split 1 F1	Sub-split 1 Exact	Sub-split 2 F1	Sub-split 2 Exact
BERT	1	89.75	89.40	89.03	86.50
	2	90.71	90.66	90.32	88.09
	3	91.12	91.04	90.28	88.02
BERT + $\mathcal{L}_{num}$	1	**90.23**	90.16	**89.90**	87.26
	2	**91.05**	90.92	**90.49**	88.52
	3	**91.46**	91.29	**91.40**	89.15

Table 5: F1 scores of BERT-base model on SQuAD 1.1 sub-splits (all scores are statistically significant with a variance of 0.01). **Sub-split 1**: both training and testing splits contains only numerical answers; **Sub-split 2**: train split contains atleast one number in the answer and testing split contains only numerical answers.

During our experiments, we observed the potential issue of weak signals from the loss when the availability of numerical pairs is sparse. In the future, our efforts would be to overcome this issue to ensure further gains.

6 Conclusion

In this work, we methodologically assign and learn embeddings for numbers to reflect their numerical properties. We validate our proposed approach with several experiments that test number embeddings. The tests that evaluate the numeral embeddings are fundamentally applicable to all real numbers. Finally, we introduced an approach to jointly learn embeddings of numbers and words that preserve numerical properties and evaluated them on a contextual word embedding based model. In our future

work, we would like to extend this idea to unseen numbers in vocabulary as a function of seen ones.

Acknowledgments

The authors would like to thank Aakanksha Naik for her help in the early stages of this work, and the anonymous reviewers as well for their insightful comments.

References

LE Blumenson. 1960. A derivation of n-dimensional spherical coordinates. *The American Mathematical Monthly*, 67(1):63–66.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pages 577–585.

Marie-Catherine De Marneffe, Anna N Rafferty, and Christopher D Manning. 2008. Finding contradictions in text. In *Proceedings of ACL-08: HLT*, pages 1039–1047.

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*.

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*.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

ZS Harris. 1954. Distributional structure. word, 10 (2-3): 146–162. reprinted in fodor, j. a and katz, jj (eds.), readings in the philosophy of language.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Chengyue Jiang, Zhonglin Nian, Kaihao Guo, Shanbo Chu, Yinggong Zhao, Libin Shen, Haofen Wang, and Kewei Tu. 2019. Learning numeral embeddings. *arXiv preprint arXiv:2001.00003*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Tom Kocmi and Ondřej Bojar. 2017. An exploration of word embedding initialization in deep-learning tasks. *arXiv preprint arXiv:1711.09160*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2009. Language models based on semantic composition. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 430–439. 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.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

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

David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Dinghan Shen, Pengyu Cheng, Dhanasekar Sundararaman, Xinyuan Zhang, Qian Yang, Meng Tang, Asli Celikyilmaz, and Lawrence Carin. 2019. Learning compressed sentence representations for on-device text processing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 107–116.

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

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. *arXiv preprint arXiv:1909.07940*.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision rnns for language recognition. *arXiv preprint arXiv:1805.04908*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*.

A Training details for Magnitude Classification Experiment

The Bi-LSTM with attention model initialized with DICE embeddings were trained on the market comments data. The model was trained for a fixed number of 9 epochs. We found that the micro and macro F1 scores peaked for a certain epoch and then flattened out. We picked the best micro and macro pair the model obtained in that certain epoch.

B Hyperparameter for BERT + $\mathcal{L}_{num}$

Our model involves a regularization method (an auxiliary loss) that can be adopted in the fine-tuning of BERT. This loss was finetuned with a hyperparameter λ and added to the existing BERT classification loss for detecting the correct span. The hyperparameter search space is between $0, 1$. We sweeped through the values manually within the search space and found that the best model that gave the maximum improvement in F1 scores had a hyperparameter value of 10^{-3}. The values were sweeped based on the observed performance. The performance faded as the hyperparameter was set to a higher value (closer to 1).

C Examples for BERT vs. BERT + $\mathcal{L}_{num}$

Figure 3 provides additional samples where BERT + $\mathcal{L}_{num}$ outperformed the baseline BERT model. Similar to previous observations, our regularized approach is able to pinpoint the correct number as opposed to selecting a substring via pattern matching.

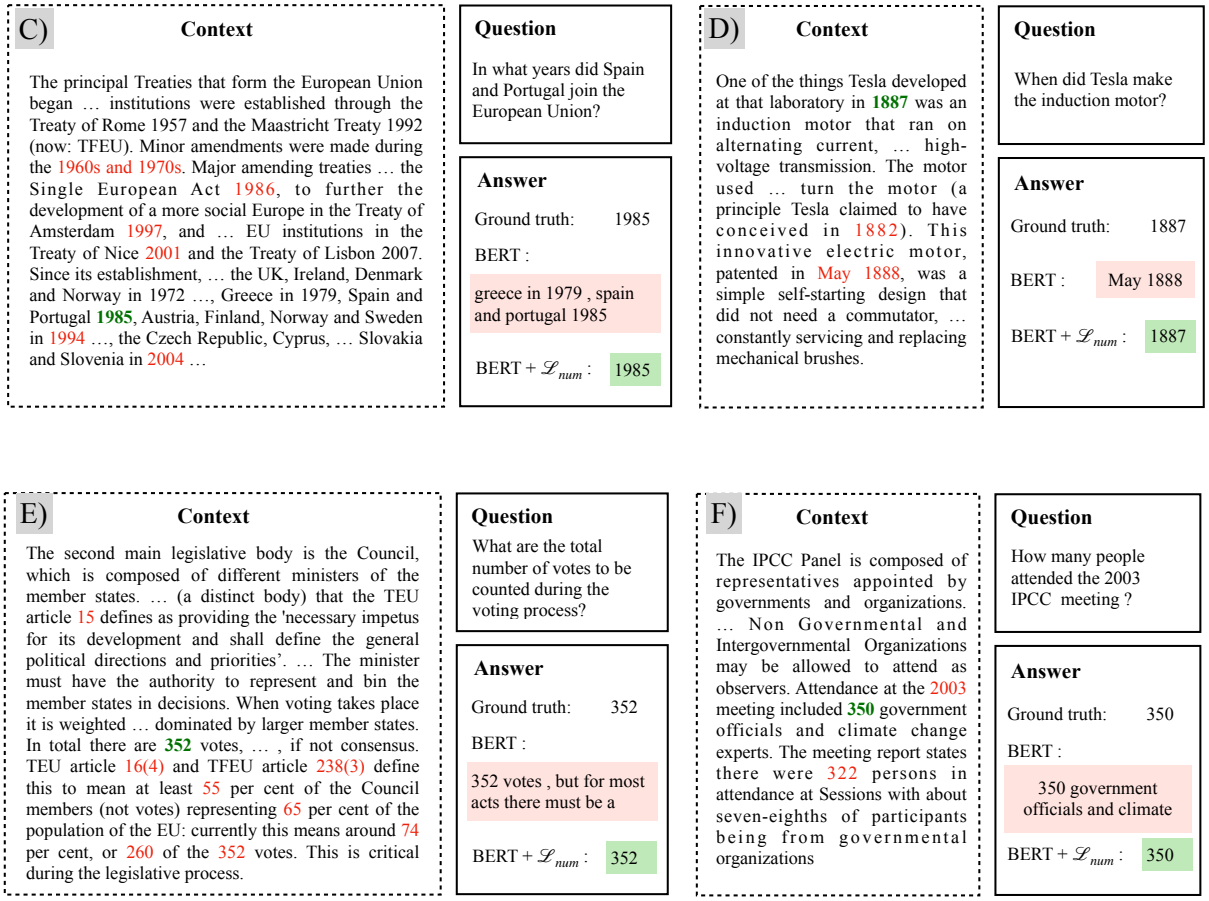


Figure 3: Qualitative examples where BERT + $\mathcal{L}_{num}$ performed better than BERT base.