Do NLP Models Know Numbers? Probing Numeracy in Embeddings

Eric Wallace*[1], Yizhong Wang*[2], Sujian Li[2], Sameer Singh[3], Matt Gardner[1]
[1]Allen Institute for Artificial Intelligence
[2]Peking University
[3]University of California, Irvine
{ericw, mattg}@allenai.org, {yizhong, lisujian}@pku.edu.cn, sameer@uci.edu

- Currently, most NLP models treat numbers in text in the same way as other tokens—they embed them as distributed vectors

- To understand how this capability emerges, we probe token embedding methods (e.g., BERT, GloVe) on synthetic list maximum, number decoding, and addition tasks.
- We repeat our probing tasks and test for model extrapolation, finding that neural models struggle to predict numbers outside the training range.

Numeracy Case Study: DROP QA

- NAQANet Model
  - Passage span
  - Question span
  - Arithmetic expression
    - In this way, we get an arithmetic expression composed of signed numbers, which can be evaluated to give the final answer.
  - Answer type

Question Type	Example	Reasoning Required
Comparative (Binary)	Which country is a bigger exporter, Brazil or Uruguay?	Binary Comparison
Comparative (Non-binary)	Which player had a touchdown longer than 20 yards?	Greater Than
Superlative (Number)	How many yards was the shortest field goal?	List Minimum
Superlative (Span)	Who kicked the longest field goal?	Argmax

Table 1: We focus on DROP *Comparative* and *Superlative* questions which test NAQANet's numeracy.

Question Type	Count	EM	F1
Human (Test Set)	9622	92.4	96.0
Full Validation	9536	46.2	49.2
Number Answers	5842	44.3	44.4
Comparative	704	73.6	76.4
Binary (either-or)	477	86.0	89.0
Non-binary	227	47.6	49.8
Superlative Questions	861	64.6	67.7
Number Answers	475	68.8	69.2
Span Answers	380	59.7	66.3

Table 2: NAQANet achieves higher accuracy on questions that require numerical reasoning (*Superlative* and *Comparative*) than on standard validation questions. Human performance is reported from Dua et al. (2019).

Stress Testing NAQANet's Numeracy

- We test two phenomena: larger numbers and word-form numbers.

Stress Test Dataset	All Questions F1	All Questions Δ	Superlative F1	Superlative Δ
Original Validation Set	49.2	-	67.7	-
Add [1, 20]	47.7	-1.5	64.1	-3.6
Add [21, 100]	41.4	-7.8	40.4	-27.3
Multiply [2, 10]	41.1	-8.1	39.3	-28.4
Multiply [11, 100]	38.8	-10.4	32.0	-35.7
Digits to Words [0, 20]	45.5	-3.7	63.8	-3.9
Digits to Words [21, 100]	41.9	-7.3	46.1	-21.6

Table 3: We stress test NAQANet's numeracy by manipulating the numbers in the validation paragraphs. *Add* or *Multiply [x, y]* indicates adding or multiplying all of the numbers in the passage by a random integer in the range [x, y]. *Digits → Words [x, y]* converts all integers in the passage within the range [x, y] to their corresponding word form (e.g., "75" → "seventy-five").

Whence this behavior?

- NAQANet demonstrates that a model can learn comparison algorithms while simultaneously learning to read and comprehend, even with only question-answer supervision.

- How, then, does NAQANet know numeracy?
  - the character-level convolutions
  - GloVe embeddings of the NAQANet model.

Probing Numeracy of Embeddings

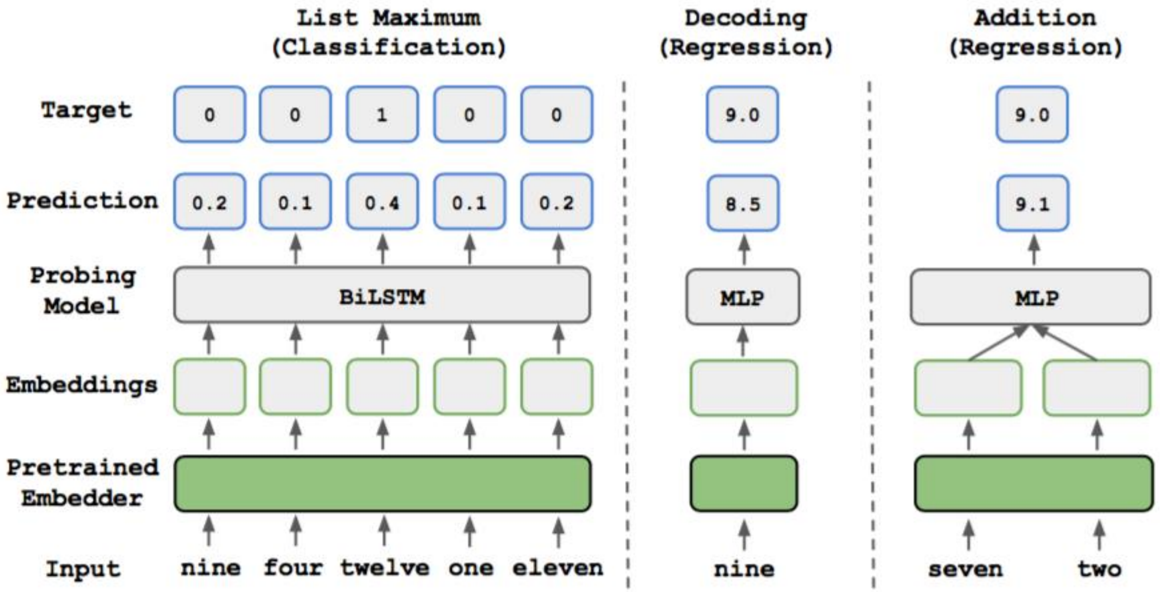

Figure 3: Our probing setup. We pass numbers through a pre-trained embedder (e.g., BERT, GloVe) and train a probing model to solve numerical tasks such as finding a list's maximum, decoding a number, or adding two numbers. If the probing model generalizes to held-out numbers, the pre-trained embeddings must contain numerical information. We provide numbers as either words (shown here), digits ("9"), floats ("9.1"), or negatives ("-9").

- the model is tested on values that are within the training range.
- We then split 80% of the numbers into a training set and 20% into a test set.

Embedding Methods

Interpolation	List Maximum (5-classes)			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
Pre-trained									
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
- GloVe	0.88	0.90	0.82	2.87	5.34	35.39	1.45	9.91	60.70

Table 4: *Interpolation with integers (e.g., "18").* All pre-trained embedding methods (e.g., GloVe and ELMo) surprisingly capture numeracy. The probing model is trained on a randomly shuffled 80% of the *Integer Range* and tested on the remaining 20%. The probing model architecture and train/test splits are equivalent across all embeddings. We show the mean over 5 random shuffles (standard deviation in Appendix D).

- All pre-trained embeddings (all methods except the Char-CNN and Char-LSTM) are fixed during training.

Results Embeddings Capture Numeracy

- Word vectors Succeed
- Character-level Methods Dominate
  - This is reflected in our probing results: character-level CNNs are the best architecture for capturing numeracy.
- Sub-word Models Struggle
  - BERT struggle
  - We suspect this results from sub- word pieces being a poor method to encode digits: two numbers which are similar in value can have very different sub-word divisions.

- A Linear Subspace Exists
  - For small ranges on the decoding task

- Value Embedding Fails

Extrapolation	**List Maximum (5-classes)**		
Test Range	[151,160]	[151,180]	[151,200]
Rand. Vectors	0.17	0.22	0.15
Untrained CNN	0.80	0.47	0.41
Pre-trained			
Word2Vec	0.14	0.16	0.11
GloVe	0.19	0.17	0.21
ELMo	0.65	0.57	0.38
BERT	0.35	0.11	0.14
Learned			
Char-CNN	0.81	0.75	0.73
Char-LSTM	**0.88**	**0.84**	**0.82**
DROP			
NAQANet	0.31	0.29	0.25
- GloVe	0.58	0.53	0.48

Table 7: *Extrapolation on list maximum.* The probing model is trained on the integer range [0,150] and evaluated on integers from the *Test Range*. The probing model struggles to extrapolate when trained on the pre-trained embeddings.

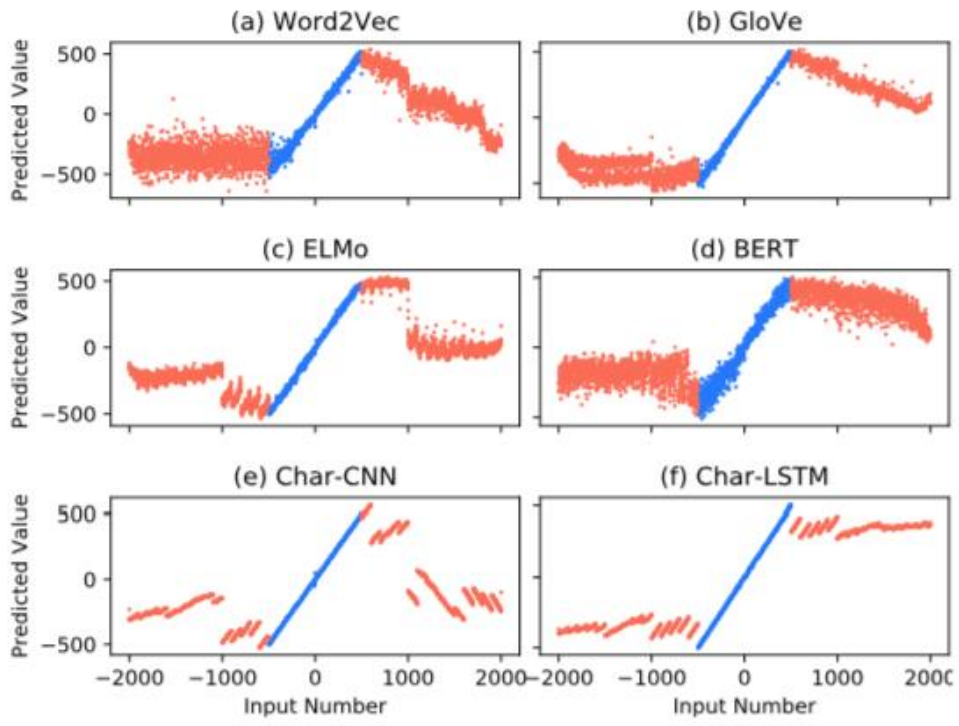

Figure 1: We train a probing model to decode a number from its word embedding over a random 80% of the integers from [-500, 500], e.g., “71” → 71.0. We plot the model’s predictions for all numbers from [-2000, 2000]. The model accurately decodes numbers within the training range (in blue), i.e., pre-trained embeddings like GloVe and BERT capture numeracy. However, the probe fails to extrapolate to larger numbers (in red). The Char-CNN (e) and Char-LSTM (f) are trained jointly with the probing model.

- we discover that pre-trained token representations naturally encode numeracy.

- it is difficult for neural models to extrapolate beyond the values seen during training

- 做一些探究性实验能不能发现一些问题
  - 比如简单的改变数字
  - 改变运算符号
  - 改变关键词