Numeracy enhances the Literacy of Language Models

Avijit Thawani **Jay Pujara** **Filip Ilievski**
University of Southern California / Los Angeles
Information Sciences Institute / Marina del Rey
{thawani,jpujara,ilievski}@isi.edu

Abstract

Specialized number representations in NLP have shown improvements on numerical reasoning tasks like arithmetic word problems and masked number prediction. But humans also use numeracy to make better sense of world concepts, e.g., you can seat 5 people in your *room* but not 500. Does a better grasp of numbers improve a model's understanding of other concepts and words? This paper studies the effect of using six different number encoders on the task of masked word prediction (MWP), as a proxy for evaluating literacy. To support this investigation, we develop Wiki-Convert, a 900,000 sentence dataset annotated with numbers and units, to avoid conflating nominal and ordinal number occurrences. We find a significant improvement in MWP for sentences containing numbers, that exponent embeddings are the best number encoders, yielding over 2 points jump in prediction accuracy over a BERT baseline, and that these enhanced literacy skills also generalize to contexts without annotated numbers. We release all code at https://git.io/JuZXn.

1 Introduction

Numbers account for 6.15% of all unique tokens in English Wikipedia (Jiang et al., 2020), yet NLP systems have traditionally either removed numbers during preprocessing or replaced them with a single uninformative UNK token. Recent models such as BERT retain them but learn individual token embeddings for hundreds of numbers. Moreover, subword tokenization approaches end up segmenting numbers into possibly suboptimal splits, e.g., 4500 is seen as (4, 500) or (45, 00) depending on the specific tokenizer used.

The human brain, in contrast, automatically maps numbers to their approximate magnitude on the number line (Dehaene, 2011). NLP systems that fail to account for the scalar values that numbers denote may correspondingly lack in comprehension. Recent work has empirically demonstrated the inefficacy of existing NLP methods in numeric reasoning tasks (Wallace et al., 2019). Alternative number representations have been proposed, such as projecting the number's magnitude into a vector space (Sundararaman et al., 2020) or switching to a scientific notation (Zhang et al., 2020; Berg-Kirkpatrick and Spokoyny, 2020).

	BERT	Exp
The [mask] weighs **100** lbs.	statue	bomb
The [mask] weighs **10000** lbs.	statue	car

Table 1: Numerate language models perform better at masked word prediction. BERT: Default BERT baseline. Exp: BERT with exponent embeddings (§2).

We observe that this line of work goes from literacy to numeracy, i.e., helping language models gain numerate skills such as simple arithmetic (Geva et al., 2020), measurement estimation (Zhang et al., 2020), and masked number prediction (Berg-Kirkpatrick and Spokoyny, 2020). Our work addresses the converse question: *Do alternative number representations enhance the ability of language models to understand/predict words?*

We investigate this question through experiments with several representative number encoders, proposed in prior work. We develop and release Wiki-Convert, a large, novel dataset of number-annotated sentences, which helps us disentangle the nominal occurrences of numbers. Our experiments show the positive impact of numeracy on a language model's literacy, as illustrated in Table 1. The default BERT model is unable to update its predictions for an object whose weight is switched from 100 to 10,000. However, our numeracy-aware method is able to predict that 100 lbs is a typical weight of a bomb, while 10,000 lbs is that of a car, due to its understanding of magnitudes and their association with words. We also find this improved *literacy* in contexts without numbers.

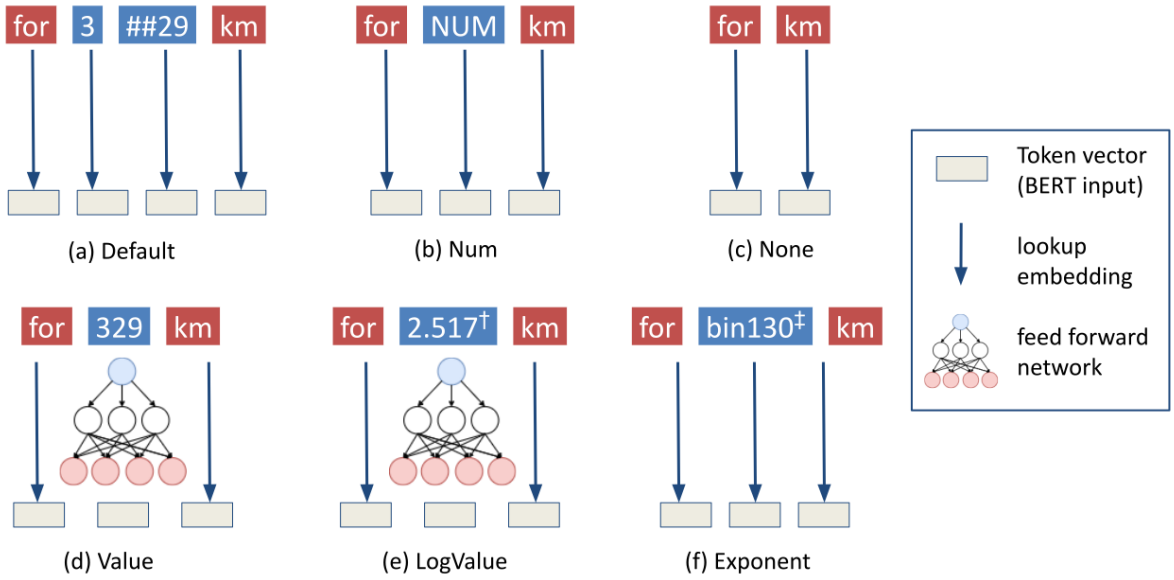

Figure 1: Different number encoders as described in Section 2. Notes: † 2.517 is $\log_{10} 329$. ‡ 329 collapses to the 130^{th} bin out of 200 log-scaled bins within our range of $[1e-4, 1e+6]$.

We promise the following contributions:

1. We are the first to show the gain from numeracy to literacy: specialized number encoders help language models better predict words.
2. We release Wiki-Convert, a new dataset containing 900k+ number-annotated sentences.

2 Methods

Our hypothesis is that language models will benefit from specialized encoders which explicitly make use of the number's magnitude. In line with both cognitive science research (Dehaene, 2011) as well as recent work on numeric representations within NLP, we propose that numbers and words be encoded differently by a language model. Words can continue to be subword-tokenized and encoded via lookup embeddings, but number encoding should consider the magnitude. We consider three representative methods from prior work which make use of a number's magnitude to encode it in vector space, as well as three baselines (marked with *), each of which is depicted pictorially in Figure 1.

1. Value embeddings (Wallace et al., 2019) project the scalar magnitude of the number to be encoded into a vector space of same dimensionality as the lookup word embeddings. We use a 1-hidden layer feed forward layer as the projection network, with a configurable number of hidden neurons.

2. LogValue is the log-scaled extension of Value, wherein the projection of the scalars is preceded by a $\log(\cdot)$ function (Wallace et al., 2019).

3. Exp or Exponent embeddings are lookup matrices for the exponent part of a scientific number notation, e.g., 2 in $3.29e2$ (Berg-Kirkpatrick and Spokoyny, 2020). Note how this method collapses numbers into equally spaced *bins* on the log scale. Although the authors used a specific implementation based on decimal scientific notation, we generalize this method to an arbitrary number of bins.

4. Default* is the usual way that BERT (Devlin et al., 2019) encode numbers: subword tokenization (Schuster and Nakajima, 2012) followed by lookup embeddings.

5. None* removes all numbers from the sentence during preprocessing. This is analogous to the baseline implementation in Berg-Kirkpatrick and Spokoyny (2020), except they mask the numbers instead of filtering them out.

6. Num* method learns a single lookup embedding for all numbers, reflecting how traditional NLP replaced any number occurrence with a single token (Graff et al., 2003), such as UNK or NUM. This method can be seen as exponent embeddings with a single bin, into which all numbers are collapsed.

3 Wiki-Convert

Existing benchmarks for numeric language modelling have been extracted automatically using regular expressions (Spithourakis and Riedel, 2018; Berg-Kirkpatrick and Spokoyny, 2020; Chen et al., 2019), and hence have no mechanism to filter out nominal numbers, such as zip codes, phone num-

Sentence	Number	Unit
U-559 had a displacement of [NUM] [UNIT] while submerged	871.0	tonne
... temperature ranges from ... in January to [NUM] [UNIT] in July	73.9	°F

Table 2: Examples sentences from Wiki-Convert along with annotated numbers and units.

bers, or proper nouns (e.g., "Boeing 747"). To allow for a more meaningful comparison, we propose *Wiki-Convert*, a novel benchmark for numeric language modeling extracted from English Wikipedia.

Wiki-Convert consists of a curated set of sentences where the numbers are not extracted by regex matching, but annotated by humans, i.e., the editors who wrote the Wikipedia article in the first place. Specifically, we make use of Convert,[1] a template that contributors have used over 3.2 million times in Wikipedia to seamlessly convert between different units of measurement. For example, {{Convert|50|mi|km}} is parsed in Wikipedia as *50 miles (80 kilometers)*. Concretely, we extract over 3 million Convert occurrences in over 1 million sentences from the May 2020 dump of English Wikipedia. We preprocess them, retaining only the 30 most frequent units (e.g., miles, acres, pounds), and filter out sentences with multiple number annotations. The end result is a dataset of over 900,000 sentences along with an annotated <number-unit> tuple. We believe Wiki-Convert can be a useful benchmark not only for numeric language modelling but also for measurement estimation tasks (Zhang et al., 2020; Zhou et al., 2020). Example Wiki-Convert annotations are shown in Table 2.

4 Experiments

We operationalize our research question by finetuning the same pretrained masked language model (BERT-base-uncased) with each of the six encoding methods (Section 2) on the task of masked word prediction. Thus when we say *numeracy*, we refer to the ability of the three number-specific encoders to take into account a number's magnitude and not its surface form. And when we say *literacy*, we refer to the masked word prediction ability of a language model, assuming it to be a valid proxy for downstream performance on other literacy tasks.

The methods encode annotated numbers into 768-dimensional vectors. Words, as well as numbers which are not annotated, are encoded by the usual subword tokenization followed by lookup embeddings. Value and LogValue methods each have a single hidden layer with 200 neurons, and exponent embeddings have 200 bins. We manually tuned the hyperparameter N (for Value, LogValue, and Exp) by optimizing for validation NLL loss over hundreds of runs, N $\in \{25, 50, 75, 100, 200, 400\}$.

Besides Wiki-Convert, we also train and test our methods on Numeracy600K (Chen et al., 2019), a dataset with financial market comments. For both datasets, we train on 100k samples, test on 10k, and use another 10k held-out dev set for configuring hyperparameters. For every input sentence, we randomly mask 15% of its non-number tokens and use a negative log likelihood loss to optimize the classifier. We measure perplexity and hit@k, masking one (non-number) word at a time.

Implementation Details We use HuggingFace Transformers (Wolf et al., 2020) for pretrained models and PyTorch Lightning (Falcon et al., 2019) for finetuning. We only train the masked language modeling (MLM) classifier (initialized from scratch) and the number encoder's parameters, if any, while keeping the base transformer weights frozen. The MLM classifier has a dense layer (768×768 weights) and a decoder (768×30522 weights) to learn output embeddings for each vocabulary item, where 768 is the embedding size for BERT-base-uncased. Value and Log Value methods with 200 hidden neurons thus consist of 768×200 (weight) + 200 (bias) + 200×1 (weight) extra parameters. Exponent embeddings with 200 bins consist of 768×200 extra parameters for the lookup embeddings. The Num model has a single lookup embedding, i.e., 768 extra parameters. None and Default methods do not contain extra parameters.

A dropout of 0.2 was applied to the hidden layer in Value and LogValue. We do not incorporate the next sentence prediction loss. While evaluating models that perplexity scores for masked LMs are lower than those of causal LMs since the former use bidirectional context while the latter only see preceding words. Our compute resources include sets of four GeForce RTX 2080 Ti GPUs, which

[1] https://wikipedia.org/wiki/Help:Convert

	Wiki-Convert					Numeracy600K				
	PPL↓	Hit@1	Hit@5	Hit@20	Hit@100	PPL↓	Hit@1	Hit@5	Hit@20	Hit@100
Default	3.11	65.57	83.86	90.97	95.75	5.07	57.94	73.52	82.84	90.53
Num	3.32	64.04	82.83	90.37	95.28	5.29	57.36	73.24	82.18	90.24
None	3.36	63.73	82.58	90.20	95.28	5.18	57.75	73.78	82.76	90.41
Value	3.28	64.54	83.01	90.51	95.37	4.90	58.90	74.37	83.62	90.66
LValue	3.26	64.67	83.07	90.48	95.43	4.90	58.66	74.68	83.54	90.73
Exp	**3.05**	**66.15**	**84.07**	**91.16**	**95.86**	**4.63**	**60.03**	**75.61**	**84.38**	**91.06**

Table 3: Results on masked word prediction over two datasets and six methods, averaged over two runs with different random seeds. PPL = Perplexity. LValue = Log Value. Exp = Exponent embeddings.

take less than twenty minutes to train a model for 10 epochs of 10k training samples (batch size 256 with accumulated gradients over 4 batches). We set batch size as 1024, the largest that we could fit onto a single GPU, since we find that large batch sizes consistently help all methods and baselines. We train all models for 10 epochs over a training set of $100k$ sentences, i.e., ~ 1000 updates, since we find this regime to allow nearly all runs to converge.

5 Results and Discussion

Our experiments help us answer three key questions about the effect of numeracy on literacy for language models:

Does numeracy help improve word prediction when numbers are present? Table 3 shows the perplexities and prediction accuracies as hit@$\{1, 5, 20, 100\}$ scores over the test splits of Wiki-Convert and Numeracy600K. We find that exponent embeddings are the top scorers on all dataset-metric combinations, achieving statistically significant improvements (at 99% confidence) against the default baseline; see Appendix A for more details. Numeracy600K is sourced from financial domain-specific articles and market comments, hence is the more challenging dataset. This is evident by the consistently higher perplexities and lower prediction scores. The Value and Log-Value methods also manage to outperform the default baseline for Numeracy600K but they score below this baseline for Wiki-Convert. However, the latter dataset was sourced from Wikipedia, over which BERT was pretrained using the default scheme, hence this makes for an unfair comparison.

	Default	Exp
PPL	7.30 ± 0.14	**7.05 ± 0.06**
H@1	51.12 ± 0.35	**51.66 ± 0.26**
H@5	67.89 ± 0.33	**68.36 ± 0.11**
H@20	78.09 ± 0.17	**78.59 ± 0.15**
H@100	86.98 ± 0.17	**87.27 ± 0.11**

Table 4: Results on masked word prediction in non-numeric contexts from Wikicorpus, averaged over three runs. PPL = Perplexity. Exp = Exponent embeddings.

Does numeracy lead to better literacy, even in contexts without numbers? We compare exponent embeddings (the best performer) against the default baseline on 1000 sampled sentences from the 2006 English dump of Wikicorpus (Reese et al., 2010) which do not have any annotated numbers. Table 4 shows that exponent embeddings continue to show much better results over the baseline.

Where exactly does numeracy help in improving literacy? We analyse examples where predictions from the default baseline erred while those from exponent embeddings were correct. Table 5 shows two representative kinds of such cases. The first three rows are examples of where we expect number encoders to help. The last row highlights a much more subtle semantic distinction (*elevation* vs *altitude*) between the two predictions. Our qualitative analysis suggests that most errors made by the default LMs are due to semantic subtleties.

Quantitatively, we further stratify our results by the kind of masked token: is it a unit (e.g., third row in Table 5) or not? Table 6 compares exponent embeddings against the default baseline, stratified over two categories of masked tokens: units and others. We find exponent embeddings to consistently outperform the default baseline over both categories. The majority of gains stem from non-unit tokens since they are more abundant than units.

The consistency of results over different corpora, configurations, and random seeds, suggest that specialized encoders do improve literacy. Such results warrant experiments on a larger scale, such as pretraining numerate language models from scratch.

Case	Sentence	Exp*	Default
Intuitive	The four petals are about **2** [mask] long and slightly hairy.	mm	meters
Intuitive	... once [mask] along the hard shoulder of the M11 at **140** mph to avoid traffic?	drove	walked
Intuitive	With its solar panels fully extended it spanned **20** [mask] .	meters	kilometers
Subtle	The Grimsel Pass is a mountain pass in Switzerland ... at an [mask] of **2164** meters.	elevation	altitude

Table 5: Qualitative error analysis over Wiki-Convert, showing examples where the **Default** baseline fails and the **Exp**onent embeddings correctly predict the masked word. Asterisk* indicates: same as ground truth.

	Exp		Default	
	Others	**Units**	**Others**	**Units**
N	7187	717	7167	716
PPL	5.22	1.76	5.38	1.95
H@1	52.80	71.06	51.60	72.28
H@5	74.55	98.44	73.85	96.88
H@20	84.98	99.87	84.74	99.09
H@100	92.78	100.00	92.45	99.87

Table 6: Results over a sample of Wiki-Convert test set, stratified by the kind of token masked.

6 Related Work

Recent NLP work has addressed several aspects of numeracy (see Thawani et al. (2021) for a recent survey). Here we review relevant prior work only on number encoders, as opposed to decoders.

Spithourakis and Riedel (2018) train a language model to predict both masked words and numbers, where the masked word prediction is the same setup as ours, while masked number prediction is modeled as a regression-penalized task of approximately estimating the number. They experiment with several number decoders, such as Digit-RNN and Gaussian Mixture Models, yet always encode numbers using a Digit-RNN, whereas we experiment with six different number encoders to evaluate their relative performance on predicting words.

Berg-Kirkpatrick and Spokoyny (2020), akin to us, employ different number encoders, but for the task of masked number prediction. Given a sentence with two numbers, they mask one number and study the effects of different number encoders for representing the unmasked number, on the task of approximately estimating the masked number.

Jiang et al. (2020) train numeral embeddings along with word vectors in a skip-gram setup, using multiple number encoders. They show that the simultaneously learned word embeddings score competitively on intrinsic word similarity tasks. Our work focuses on this literacy evaluation, revealing that some number encoders not only help language models perform at par with the default baseline, but also exceed it in terms of perplexity.

Lastly, Zhang et al. (2020) pretrain BERT-base with a modified training corpus where all numbers are converted to scientific notation. This variant of BERT, called NumBERT, converges to a similar loss on masked language modeling and next sentence prediction objectives as BERT-base. On language understanding tasks like NLI, NumBERT is only a little worse than BERT-base. They also employ a LogValue *decoder* (which they call RGR) to probe NumBERT embeddings on the task of measurement estimation. Our work instead focuses on word prediction, but our results converge in showing that better numeric representations do not harm (instead improve) language modelling abilities.

7 Conclusion

Our work studies the effect of number encoders on the task of masked word prediction, as a proxy for the ability of understanding text. We show that specialized number encoders are helpful in improving the *word* prediction ability of a language model, evaluated by perplexity and hit@k scores. We demonstrate these gains not only over sentences with annotated numbers but also more generally on text without numbers. We find exponent embeddings to be the best number encoders for masked word prediction. We see our work as preliminary evidence that numeracy enhances the literacy of language models. To facilitate subsequent work, we develop and release Wiki-Convert: a novel resource for number-related NLP with the added advantage of not conflating nominal with ordinal numbers.

Future Work: We observe that the best performing number encoder (Exp) is not merely magnitude-aware (so are Value and LogValue) but is also learnt by lookup embeddings over collapsed numbers ranges. So far we explicitly define these ranges as those on the log scale but we intend to explore data-driven methods of identifying ranges from raw text, e.g., 1939-45 as the range of years of WW2.

8 Acknowledgements

This work was funded by the Defense Advanced Research Projects Agency with award N660011924033. We would like to thank the anonymous ACL 2021 and EMNLP 2021 reviewers for helping us refine earlier versions of this paper.

9 Ethical Considerations

Our work exclusively revolves around the English language and the Hindu-Arabic Numeral system, which are by no means the only language / number systems in use today. We encourage follow-up work to take other systems into consideration, on the lines of Johnson et al. (2020) and Nefedov (2020). Wiki-Convert, our dataset has been extracted from Wikipedia dumps, which are licensed under the GNU Free Documentation License (GFDL) and the Creative Commons Attribution-Share-Alike 3.0 License. The authors were in no way involved in the annotation process, which is contributed to by volunteer Wiki editors making use of the Convert template. We note, however, that the units of measurement we filter out (due to rare occurrences) will cause a cultural bias towards European and American units, such as pounds or miles, since they are over-represented in English Wikipedia. As a remedy, we shall release extraction scripts to enable researchers in creating other versions of Wiki-Convert, perhaps even supporting multiple languages.

References

Taylor Berg-Kirkpatrick and Daniel Spokoyny. 2020. An empirical investigation of contextualized number prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4754–4764, Online. Association for Computational Linguistics.

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. 2011. *The number sense: How the mind creates mathematics*. OUP USA.

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.

William Falcon et al. 2019. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, Online. Association for Computational Linguistics.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34.

Chengyue Jiang, Zhonglin Nian, Kaihao Guo, Shanbo Chu, Yinggong Zhao, Libin Shen, and Kewei Tu. 2020. Learning numeral embedding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2586–2599, Online. Association for Computational Linguistics.

Devin Johnson, Denise Mak, Andrew Barker, and Lexi Loessberg-Zahl. 2020. Probing for multilingual numerical understanding in transformer-based language models. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 184–192, Online. Association for Computational Linguistics.

Mikhail Nefedov. 2020. Dataset for evaluation of mathematical reasoning abilities in russian. In *Conference on Artificial Intelligence and Natural Language*, pages 135–144. Springer.

Samuel Reese, Gemma Boleda, Montse Cuadros, Lluís Padró, and German Rigau. 2010. Wikicorpus: A word-sense disambiguated multilingual Wikipedia corpus. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Georgios P. Spithourakis and Sebastian Riedel. 2018. Numeracy for language models: Evaluating and improving their ability to predict numbers. *CoRR*, abs/1805.08154.

Dhanasekar Sundararaman, Shijing Si, Vivek Subramanian, Guoyin Wang, Devamanyu Hazarika, and Lawrence Carin. 2020. Methods for numeracy-preserving word embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4742–4753, Online. Association for Computational Linguistics.

Avijit Thawani, Jay Pujara, Pedro A. Szekely, and Filip Ilievski. 2021. Representing numbers in NLP: a survey and a vision. *CoRR*, abs/2103.13136.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xikun Zhang, Deepak Ramachandran, Ian Tenney, Yanai Elazar, and Dan Roth. 2020. Do language embeddings capture scales? In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4889–4896, Online. Association for Computational Linguistics.

Ben Zhou, Qiang Ning, Daniel Khashabi, and Dan Roth. 2020. Temporal common sense acquisition with minimal supervision. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7579–7589, Online. Association for Computational Linguistics.

A Significance Tests

We attempted multiple kinds of significance tests for our experiments comparing exponent embeddings with the default baseline over Wiki-Convert test set, each with a different set of assumptions:

Unpaired Student's T-test: We assume each masked token prediction to be a binary sample, marked as 1 if correct, else 0. Exponent embeddings accuracy scores are significantly better than default baseline with p value approximately 2^{-6}. Note that we do not use a paired t-test since the total number of available samples or masked tokens differ slightly (163942 for exponent and 163873 for default) due to the subword tokenization followed by the default baseline.

Binomial Distribution: Here we assume each masked word prediction as a binomial draw, with base probability equal to the overall accuracy of the default baseline. We find that the probability of a method merely as good as the default baseline obtaining the overall accuracy obtained by exponent embeddings (or more) is below 1^{-11}, or the p-value is 1^{-13}, thereby demonstrating statistical significance.

Bootstrapping: This is the least presumptive of statistical tests. Over 10K bootstrapped runs, we find that the default and exponent Hit@1 scores translate to 99% confidence intervals of [65.22, 65.79] and [65.96, 66.62] respectively, which are non-overlapping, hence the difference is statistically significant.