

Appendices: Improving Low Compute Language Modeling with In-Domain Embedding Initialisation

Charles Welch, Rada Mihalcea and Jonathan K. Kummerfeld
















Computer Science & Engineering

University of Michigan

{cfwelch, mihalcea, jkummerf}@umich.edu

A Initialisation without additional data

This experiment considers a variation where we pretrain the embeddings only on the PTB (i.e., there is no additional pretraining data). LM uses the embeddings produced by training a baseline model (i.e., train an LM, then reset all parameters except the embeddings and train again).

Tie	Input	Pretraining Method	Val PPL
		GloVe	64.5
		GloVe	66.2
		LM	61.7
			61.3
		GloVe	60.5
		LM	60.3
		LM	59.4

While pretraining on the training data improves performance here, the improvement does not persist through the finetuning stage.

B Note on model size

When we untie the embeddings it does increase the number of parameters. We ran experiments with 200 dimensional embeddings and found the same trends, but all results were worse. This indicates that the larger dimensionality is necessary.

C Reproducibility Criteria

For each item in the list we have a section below with the relevant information.

C.1 Experimental Results

A clear description of the mathematical setting, algorithm, and/or model. The model we use is described in Merity et al. (2017). We modify it to support weight freezing and initialisation.

For embeddings, we used GloVe with the same configuration as described in the original paper.

For words in the LM training set that do not appear in the pretraining data, we used a random vector generated in the same way as Merity et al. (2017).

A link to a downloadable source code, with specification of all dependencies, including external libraries The code for our work is attached as supplementary material and available at <http://jkk.name/emnlp201m/>. For the main experiments we used CUDA 10.1 and PyTorch 0.1 to get results consistent with those reported in Merity et al. (2017).¹

Description of computing infrastructure used

We used 7 GeForce GTX TITAN X GPUs with 12212 Mb of RAM each. For the GPT-2-large experiments we used a Tesla T4 via Google Colab based on the notebook from Callison-Burch and Ippolito (2020).

Average runtime for each approach Time per epoch varied from 60 to 2250 seconds depending on the amount of training data.

Number of parameters in each model For the language model this varied from 24,221,600 to 105,737,253, depending on the training data used. Those values do not count all of the pretrained vectors though, only the ones that occurred in either the training, development, or test sets. The pretrained vectors depended on the volume of data:

- Cord: 126,386,800
- IRC: 26,050,000
- NANC: 29,019,200
- Reddit: 77,719,200
- Wiki: 624,022,000
- Gigaword: 410,236,000

Corresponding validation performance for each reported test result Only the final table in the paper reports test results and it also contains the relevant validation results.

¹Later versions of PyTorch and their code led to slightly worse performance.

Explanation of evaluation metrics used, with links to code We use perplexity as implemented in (Merity et al., 2017). The code is attached and available at <http://jkk.name/emnlp201m/>.

C.2 Hyperparameter Search

We performed one hyperparameter search as described in the paper.

Bounds for each hyperparameter These values are named as defined in the AWD-LSTM arguments:

- lr, [10, 25].
- dropouti, [0, 0.5].
- dropoute, [0, 0.5].
- nhid, [650, 1650].
- nlayers, [2, 5].
- dropout, [0.25, 0.55].
- dropouth, [0.1, 0.4].
- wdrop, [0.35, 0.65].

We fixed clipping, batch size, bptt, and wdecay based on observations in prior work. For hyperparameter tuning we reduced the number of epochs to 100 as improvements beyond that point were usually very small. We also stopped early if the loss at epoch N was not lower than in epochs [0: N-10], as proposed in Merity et al. (2018).

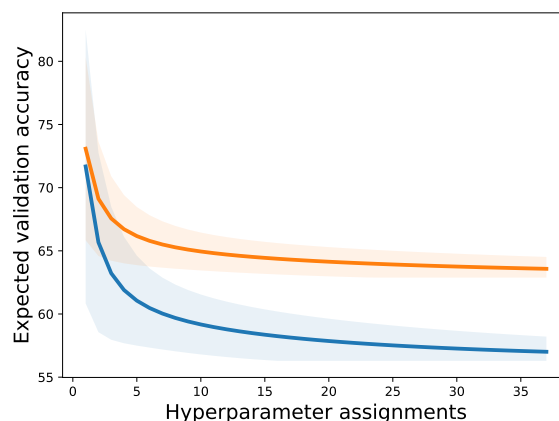
Hyperparameter configurations for best-performing models For most experiments we used the default hyperparameters from the AWD-LSTM, GPT-2 and kenlm. For the final experiments we used tuned hyperparameters as specified below.

Parameter	Tuned	Baseline
lr:	14.0	30
dropouti:	0.28	0.4
dropoute:	0.05	0.1
nhid:	1322	1150
nlayers:	4	3
dropout:	0.28	0.4
dropouth:	0.31	0.25
wdrop:	0.58	0.5

Number of hyperparameter search trials See the paper.

The method of choosing hyperparameter values (e.g., uniform sampling, manual tuning, etc.) and the criterion used to select among them (e.g., accuracy) All of the hyperparameters were simultaneously varied, sampling all uniformly at random. We selected the final set based on validation perplexity.

Expected validation performance, as introduced in Section 3.1 in Dodge et al, 2019, or another measure of the mean and variance as a function of the number of hyperparameter trials. We use Dodge et al. (2019)’s approach to produce the following plot, where the orange line is the baseline and our approach is the blue line:



C.3 Datasets

We use several datasets:

- PTB: Mikolov’s preprocessed version of the PTB (Marcus et al., 1993).
- PTB-Rare: Our version of the PTB.
- NANC: Wall Street Journal data from the North American News Corpus (Graff, 1995).
- Wiki: English Wikipedia.
- Reddit: A random sample of Reddit messages.
- CORD-19: Covid-19 research articles tokenised by Scispacy (Neumann et al., 2019).
- IRC: Ubuntu IRC dialogue disentangled by prior work (Kummerfeld et al., 2019).
- Gigaword: The Gigaword corpus (Parker et al., 2011).

Relevant statistics such as number of examples See below.

Details of train/validation/test splits

- PTB: We used the standard split: sections 00-20 for training, 21-22 for validation, 23-24 for test. These contain 88,7521, 70,390, and 78,669 tokens respectively.

- PTB-Rare: This is the same split as PTB.
- NANC: We randomly divide the data into splits the same size as PTB. For the additional LM data experiment we expand the training set first with the same amount of data again, then add another two times the data (adding the data each time, so smaller sets are subsets). The samples are not exactly the same size as the PTB training set as we add articles until we have just gone past the number of tokens. In all cases the pretraining, validation, and test set have 43,098,002, 72,356 and 79,408 tokens respectively. The 1x, 2x, and 4x training sets have 887,993, 1,776,407, and 3,551,496 tokens respectively. For the final experiment we use an 8x set (the 4x set plus another 4x) containing 7,101,988 tokens.
- Wiki: The same process as NANC. This gave pretraining, training, validation, and test sets of size 2,247,381,902, 887,933, 70,437, and 80,180 respectively.
- Reddit: The same process as NANC. This gave pretraining, training, validation, and test sets of size 219,940,812, 887,617, 70,439, and 78,796 respectively.
- CORD-19: The same process as NANC. This gave pretraining, training, validation, and test sets of size 194,840,142, 891,989, 73,014, and 81,648.
- IRC: The same process as NANC. This gave us pretraining, training, validation, and test sets of size 53,443,738, 887,716, 70,530, and 78,784.
- Gigaword: The same process as NANC for selecting 2x and 4x data. This gave us 4,790,293,007 tokens for pretraining, 888,869 extra tokens for 2x (added to the base NANC training data) and 2,664,487 extra tokens for 4x.

Explanation of any data that were excluded, and all pre-processing steps For all datasets aside from PTB and Reddit we used entire articles / conversations rather than breaking them into separate sentences. We applied preprocessing similar to the Mikolov PTB data, except that we do not remove rare words. Our scripts for preprocessing are in the attached code and at <http://jkk.name/emnlp201m/>.

- PTB: We used the raw data directly.

- PTB-Rare: All data, preprocessed with the `make-non-unk-ptb.py` script.
- NANC: We used all articles from the Wall Street Journal, concatenating lines to form complete articles. For BPE evaluation we used the GPT-2 tokenizer to prepare the data.
- Wiki: Extracted using <https://github.com/attardi/wikiextractor>, then removed text that contained 'colspan' or 'rowspan', as wikiextractor sometimes extracts parts of tables. Also excluded titles.
- Reddit: Used the Stanford CoreNLP tokenizer.
- CORD-19: We use all of the articles with pmc json data that are shorter than 20,000 tokens (99% of the data).
- IRC: We use all of the automatically disentangled conversations.
- Gigaword: We use all of the articles, removing duplicates.

A link to a downloadable version of the data
The raw data is available at the links below. The preprocessing we applied is described above.

- PTB: <http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz>
- PTB-Rare: <https://catalog.ldc.upenn.edu/LDC99T42>
- NANC: <https://catalog.ldc.upenn.edu/LDC95T21>
- Wiki: <https://dumps.wikimedia.org/backup-index.html>
- Reddit: https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment/
- CORD-19: <https://www.semanticscholar.org/cord19>
- IRC: <https://github.com/jkkummerfeld/irc-disentanglement>
- Gigaword: <https://catalog.ldc.upenn.edu/LDC2011T07>

For new data collected, a complete description of the data collection process, such as instructions to annotators and methods for quality control. We did not collect any new data.

References

- Chris Callison-Burch and Daphne Ippolito. 2020. Homework 4: Fine-tune a neural language model for text generation. <http://interactive-fiction-class.org/homeworks/language-model/language-model.html>.
- Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. **Show your work: Improved reporting of experimental results.** 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 2185–2194.
- David Graff. 1995. **North american news text corpus LDC95T21.** Web Download. Philadelphia: Linguistic Data Consortium.
- Jonathan K. Kummerfeld, Sai R. Gouravajhala, Joseph J. Peper, Vignesh Athreya, Chulaka Gunasekara, Jatin Ganhotra, Siva Sankalp Patel, Lazaros Polymenakos, and Walter S. Lasecki. 2019. **A large-scale corpus for conversation disentanglement.** In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3846–3856.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. **Building a large annotated corpus of English: The Penn Treebank.** *Computational Linguistics*, 19(2):313–330.
- Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. **Regularizing and Optimizing LSTM Language Models.** In *International Conference on Learning Representations (ICLR)*.
- Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. **An Analysis of Neural Language Modeling at Multiple Scales.** *arXiv preprint arXiv:1803.08240*.
- Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. **ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing.** In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327.
- Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. **English gigaword fifth edition LDC2011T07.** Web Download. Philadelphia: Linguistic Data Consortium.