HW2: Modeling

Justin Chiu
justinchiu@seas.harvard.edu

Demi Guo
dguo@college.harvard.edu

Yuntian Deng
dengyuntian@seas.harvard.edu

February 13, 2018

1 Introduction

Language modeling is a central building block in natural language processing which has a lot of applications, such as machine translation (Bahdanau et al., 2014; Luong et al., 2015), dialogue (Serban et al., 2016), summarization (Rush et al., 2015), and speech recognition (Hannun et al., 2014; Chan et al., 2016; Amodei et al., 2016).
In this paper, our goal is to explore different models for language modeling, which includes:

1. A traditional ngram model (unigram, bigram, trigram) (Heafield, 2011)
2. A Neural probabilistic language model (Bengio et al., 2003)
3. A basic LSTM language model with dropout (Zaremba et al., 2014)
4. Some extensions of our basic LSTM language model, such as adding the neural cache model (Grave et al., 2016)

We evaluate our language model on the Penn Tree Bank (Marcus et al. (1993)), which is a commonly-used benchmark dataset for evaluating language modeling. We report the perplexity of our models on train and validation dataset, as well as their performances on predicting the next word given previous 10 words, measured by MAP of top 20 predictions.

2 Problem Description

A statistical model of language is a probability distribution over sequences of words $w_1, w_2, \cdots, w_n$. Given such a sequence, say of length n, our goal is to assign it a probability $P(w_1, \cdots, w_n)$ to the whole sequence.
We can factor the probability of a word sequence by the conditional probability of the next word given all the previous ones:

$$P(w_{1:t}) = \prod_{t=1}^{T} P(w_t | w_{1:t-1}) \tag{1}$$

Under such factorization, we need to model $P(w_t|w_{1:t-1})$. In this paper, we use ngrams or a neural network to model this probability distribution.

3 Model and Algorithms

3.1 Ngram Model

In ngram models, we make a Markovian assumption that $P(w_t|w_{<t}) = P(w_t|w_{t-1}, w_{t-2}, \cdots, w_{t-n+1})$. For a large n value, a potential drawback of such a model might be the curse of dimensionality: the training data we need grows exponentially with n, and during inference there might be some ngrams never appearing in training corpus. Therefore, it is very common to linearly interpolate it with lower order ngram models (Heafield, 2011).

In this paper, we set $n = 3$:

$$p(y_t|y_{1:t-1}) = \alpha_1 p(y_t|y_{t-2}, y_{t-1}) + \alpha_2 p(y_t|y_{t-1}) + (1 - \alpha_1 - \alpha_2) p(y_t) \tag{2}$$

Our final best model has $\alpha_1 = 0.7, \alpha_2 = 0.2$

We also experimented with different smoothing methods. For example, we tried to use add-alpha smoothing for unigram, bigram and trigram.

Moreover, we used add-alpha smoothing for unigram model with $\alpha = 1$. For bigram and trigram, if the bigram and trigram previx didn't occur before, we will simply set the probability to be 0.

3.2 Neural Network Language Model

Our NNLM model is a slight augmentation of the one in Bengio et al. (2003), where we have a feed-forward network over a window of embeddings, and then additionally a small multilayer perceptron on top of that. We found the following hyperparameters to work well and achieved a test perplexity of 146.1:

- Adam as the optimization routine
- Learning rate of 0.001
- A window of size 3
- One addititional intermediate hidden layer between the projection over n-grams and the softmax
- Dropout with probability 0.5 after the window projection and the intermediate hidden layer

3.3 LSTM Language Model

Our LSTM language model closely resembles the baseline in Grave et al. (2016), however we additionally use weight-tying (Press and Wolf (2016)) (where we share the input embedding weights with the vertices of the softmax polytope) which decreases perplexity. We were able to match the baseline results with a test perplexity of 72.8 in Table 3 of Merity et al. (2017) with the following hyper-parameters:

- Adam as the optimization routine
- Learning rate of 0.01
- Momentum parameters set to the default [0.9, 0.999])
- Max-norm gradient clipping at 2
- Weight decay at 0.0001
- 2-layer LSTM model
- 512 hidden units for each RNN and the embeddings
- 0.5 dropout after each recurrent layer
- Weight-tying (sharing) for the embeddings and vertices of the softmax polytope

3.4 Extensions to LSTM language model

We experiment with adding a continuous cache (Grave et al. (2016)) on top of our trained LSTM model from last section.

Now, we have $p(w|h_{1...t}, x_{1...t}) = (1-\lambda)p_{vocab}(w|h_t) + \lambda p_{cache}(w|h_{1...t}, x_{1...t})$,
where $p_{cache}(w|h_{1...t}, x_{1...t}) \propto \sum_{i=1}^{t-1} 1_{w=x_{i+1}} exp(\theta h_t^T h_i)$. We search the hyperparameters based on model's validation performance.

In our best model, $\theta = 1.0$, and $\lambda = 0.1$. We stored 2000 most recent words and hidden states pairs in history.

Testing is quite different for this cahce model, since now we need to generate the next word given only last 10 words. Now, for each data point, we will add the first 9 words to our cache. We ignore the 10th word intentionally since we don't know what its next word is, which is required for the cache model. Our cache will accumulate through different data points,

Here, our cache perhaps should not be interpreted as a cache for recently occurred words, but a database. It's intuitively still useful since if we have two similar hidden states, then it may imply the next words should be the same.

3.5 Ensemble Model

Finally, we ensemble all above models by averaging the probabilities, and report its performance alongside other models.

4 Experiments

Our main results are shown in Table 1. LSTM CACHE achieves the best performance, even better than ENSEMBLE. We submitted LSTM CACHE to Kaggle and achieved **1st** place in the public ranking task. Neural models outperform Ngram models by a large margin, which is not surprising.

We observe that weight-tying can give a significant performance boost in the case of the LSTM model, but significantly hurts performance for the NNLM model.

Model	Test PPL
NGRAM INTERP (3,2,1)	241.977
NNLM	152.1
LSTM LM	72.8
LSTM CACHE	69.2
ENSEMBLE	79.6

Table 1: Test perplexities on Penn Tree Bank.

Model	Test PPL
NNLM + WEIGHT TYING	152.1
NNLM - WEIGHT TYING	208.3
LSTM + WEIGHT TYING	72.8
LSTM - WEIGHT TYING	82.2

Table 2: The effect of weight tying.

5 Conclusion

On the task of language modeling, we experimented with various models. We found that models with heavy regularization, as well as longer memory were able to achieve lower perplexities on the Penn Tree Bank dataset.

References

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pages 173–182.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE.

Grave, E., Joulin, A., and Usunier, N. (2016). Improving neural language models with a continuous cache. *CoRR*, abs/1612.04426.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al. (2014). Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*.

Heafield, K. (2011). Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Merity, S., McCann, B., and Socher, R. (2017). Revisiting activation regularization for language rnns. *CoRR*, abs/1708.01009.

Press, O. and Wolf, L. (2016). Using the output embedding to improve language models. *CoRR*, abs/1608.05859.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Zaremba, W., Sutskever, I., and Vinyals, O. (2014). Recurrent neural network regularization. *CoRR*, abs/1409.2329.