VIVO: Vector In - Vector Out
Neural Sequence Models with Continuous Outputs

Brendon Boldt
bboldt@cs.cmu.edu

Felix Labelle
flabelle@cs.cmu.edu

Artidoro Pagnoni
apagnoni@cs.cmu.edu

Abstract

We implement two neural models for machine translation: a baseline sequence-to-sequence neural model with attention [Bahdanau et al., 2015] and the continuous output neural sequence model proposed by Kumar and Tsvetkov [2019]. Along with reporting our results on both models, we analyze the continuous output model, identifying some of its weaknesses with respect to the traditional model. We propose to mitigate some of these shortcomings by jointly learning the target embedding space of the continuous output model together with the model parameters. Our implementation can be found at: `https://github.com/artidoro/vivo`.

1 Introduction

The standard approach for neural machine translation [Bahdanau et al., 2015, Vaswani et al., 2017] samples words from a categorical distribution over a fixed size vocabulary. The distribution over the vocabulary is estimated by a linear layer which computes scores for each candidate word followed by normalization through a softmax layer. This step is computationally expensive and many approaches have been proposed [Morin and Bengio, 2005, Chen et al., 2016, Sennrich et al., 2016] to reduce the time required. One particular approach is the regression of word embeddings. This report details the reimplementation of the model outlined in [Kumar and Tsvetkov, 2019], evaluates its performance, provides an error analysis, and proposes approaches for addressing the errors identified.

2 Baseline

We implement two baseline models: the standard attentional neural machine translation model [Bahdanau et al., 2015, Luong et al., 2015], and the continuous output model proposed by [Kumar and Tsvetkov, 2019] in Pytorch [Paszke et al., 2019]. The two model's architectures are identical. Both models use an encoder-decoder model with LSTMs [Hochreiter and Schmidhuber, 1997] and global attention [Luong et al., 2015] in the decoder. Following the implementation of these models by Kumar and Tsvetkov [2019] we add input feeding and at decode time we replace the unk token with the word with highest attention in the source sentence.

The main difference between the standard model and the continuous output model is that the former treats the prediction problem as a classification problem over the vocabulary, while the latter models it as a regression problem. Instead of having a projection layer to the vocabulary followed by a softmax and using cross entropy loss, the continuous output model proposed by Kumar and Tsvetkov [2019] outputs a vector directly and uses a loss function based on the von Mises-Fisher (vMF) distribution. See section 2 for a description of the computation of the loss function. The target in the softmax model is the index of the target word, while in the continuous output model, it is the embedding of the target word.

The hyperparameters used for these baselines are presented in Table 3. The values chosen were largely taken from Kumar and Tsvetkov [2019]. The loss function was also reimplemented from scratch.

2.1 Loss Calculation

The von Mises-Fisher distribution is given by the following equation:

$$p(\mathbf{e}(w); \boldsymbol{\mu}, \kappa) = C_m(\kappa) e^{\kappa \boldsymbol{\mu}^\top \mathbf{e}(w)}, \tag{1}$$

$$\text{where} \quad C_m(\kappa) = \frac{\kappa^{m/2-1}}{(2\pi)^{m/2} I_{m/2-1}(\kappa)}. \tag{2}$$

The loss function is given by the negative log likelihood of this equation specifically where $\boldsymbol{\mu}$ is the vector generated by the model scaled to have unit norm, κ is the original norm of the output vector, m is the dimension of the vectors, and $\mathbf{e}(w)$ is the normalized ground truth word vector. Our model implemented this loss function in like manner to Kumar and Tsvetkov [2019] with the exception of the normalizing constant $C_m(\kappa)$. Since $m = 300$ for our experiments, this requires computing the gradient of the modified Bessel function of the first kind ($I_{m/2-1}(\kappa)$) with order 149; this is problematic both due to numerical instability and due to the fact that the Bessel function has no closed form and is instead expressed as an infinite series.

Kumar and Tsvetkov [2019] use a closed-form lower bound approximation which provides an accurate estimation of the gradient. The lower bound of the loss function is,

$$\log(C_m(\kappa)) \geq \sqrt{(m/2)^2 + \kappa^2} - (m/2) \log\left(m/2 - 2 + \sqrt{(m/2)^2 + \kappa^2}\right) \tag{3}$$

We alternatively tried estimating the Bessel function directly by computing the first 10 terms of the series in log space. This gives a slightly noisier estimate of the gradient, but we found that it lead to better performance in our experiments and used it for the reported numbers.

2.2 Decoding

To compare with the main result of Kumar and Tsvetkov [2019], we use simple greedy decoding for both softmax and vMF models. Beam search was also implemented and is discussed in Section 4.2. In the softmax model this is done by selecting the word with highest softmax score. The vMF model on the other hand selects the word embedding closest to the model's output vector in order to generate that word. Notice that these two approaches have the same asymptotic time complexity: linear in the size of the vocabulary.

Kumar and Tsvetkov [2019] reuse the loss function for the nearest-neighbor search. Since the embedding vectors being searched over are normalized to have unit norm, it can be shown that minimizing the loss function is identical to maximizing the cosine similarity. We have used both approaches and find no difference.

2.3 Dataset & Embeddings

We experiment with three different language pairs to train and evaluate the two models: German-English, French-English, and English-French. We use the training set of IWSLT 16 dataset [Cettolo et al., 2016] for training, the evaluation set is the combination of TED TEST 2013-2014, and the test set is the combination of TED TEST 2015-2016. We follow the same approach as Kumar and Tsvetkov [2019] to be able to comapre our results. Also following Kumar and Tsvetkov [2019] sentence pairs were filtered based on their length, with pairs containing sentences above a `max_len` $= 100$ were removed. The results from each of these language pairs are presented in subsection 2.3.

The vMF model is very sensitive to the target embeddings used to compare model predictions. We observed that different initializations of some of these embeddings (end-of-sentence token, unknown word token, ...) significantly altered the results of the model. We use FastText [Bojanowski et al., 2016] to train the target embeddings. Additional French and English data from WMT 16 [Bojar et al., 2016] were used to train word embeddings. These embeddings were trained using the settings recommended by Bojanowski et al. [2016]. We calculated the embedding for the `<unk>` metatoken by taking the negative of the average of the words present embeddings but absent from the target vocabulary.

Loss	fr-en	de-en	en-fr
CE (Kumar)	32.0	24.7	29.3
CE (ours)	**32.4**	**25.9**	**34.6**
NLLvMF (Kumar)	**32.1**	**25.1**	**31.7**
NLLvMF (ours)	29.2	21.7	29.7

Table 1: BLEU scores on IWSLT'16 test set (ours versus Kumar and Tsvetkov [2019]).

3 Results

In Table 1 we report BLEU score results on the IWSLT'16 test set for language pairs German-English, French-English, and English-French. We observe that the results of the standard machine translation model using cross-entropy loss (CE) matches and improves on the results form Kumar and Tsvetkov [2019].[1] However our results on the von Mises-Fisher loss do not match the original implementation. The difference is particularly noticeable for German-English in comparison to the other language pairs.

The results that we report use the hyperparameters described in Table 3 in the appendix, which match the values provided by Kumar and Tsvetkov [2019]. We did however explore different values for hyperparameters linked to the vMF loss. In particular we tested a range of values for both regularization constants ($\lambda_1 = 0.3 - 0.07$, $\lambda_2 = 0.015 - 0.03$), and the learning rate (0.001-0.0001). We also explored different reductions for the loss (token level , and sentence level) finding that the token level reduction (averaging the loss per token) gave better results for the vMF loss. Finally we also tried two different approximations of the computation of the Bessel function used in the normalization constant of the vMF distribution. We tested the lower bound suggested by Kumar and Tsvetkov [2019] and an approximation based on finite sums. The results reported use the finite sum approximation.

An advantage of the continuous output model that was evident from our experiments is the improved training time. This is due to the reduced number of parameters that require training (the embeddings of the target language remain fixed and are not trained, and there is no projection layer to the vocabulary).

4 Analysis

Overall our implementation of the vMF model underperformed Kumar and Tsvetkov [2019]'s results. We suspect that this is due to the vMF model being very sensitive to parameter initialization and approximations to the constant term in the vMF loss. We believe that the initialization of the embedding vector for the unknown token is not optimal. We use the initialization described in Kumar and Tsvetkov [2019], which sets the vector to the negative average of all embeddings available for words that are not present in the target vocabulary. This is an arbitrary initialization which leads to the norm of the unknown token embedding to be very small (more than two standard deviation from the mean). Another noticeable difference with the model described in Kumar and Tsvetkov [2019], is that we did not use a word dictionary created through fast-align to look up results.

4.1 Error Analysis

Some errors that appeared frequently in the model outputs were word duplicates. Simple single duplicates removal leads to a 0.5 BLEU improvement in model performance for De→En. Similarly, there were n-gram level repetitions. vMF appears more susceptible to these types of errors than the traditional softmax model. Out of 50 samples observed from a test set with 2300 samples for De→En, vMF has 2 n-gram repetitions compared to 0 for softmax.

Our implementation struggled with longer sentences, often drifting in meaning towards the end of the sentence. Examples of common error types can be found in Table 4 in the appendix.

[1] Notice that our test set results still had unks, while it contained the original words for the results reported by Kumar and Tsvetkov [2019]

Loss	BS/k	BLEU	Acc.
SM	1	25.1	0.58
SM	3	26.7	0.72
SM	5	27.0	0.77
SM	10	26.8	0.81
vMF	1	23.5	0.57
vMF	3	23.8	0.64
vMF	5	24.3	0.67
vMF	10	24.2	0.70

Table 2: Beam search results for a softmax and von Mises-Fisher model on the validation set. "BS/k" refers to the beam size and top-k values. Note that these experiments were run without the replacement of <unk> metatokens.

4.2 Beam Search

Kumar and Tsvetkov [2019] mention briefly that beam search using the nearest neighbors with respect to vMF loss did not significantly improve the performance of the model. We implemented beam search whose results are presented in Table 2. This is borne out by a comparison of the improvement beam search gives to softmax-based model vs. the vMF model; namely, the softmax sees a 1.9 point improvement in BLEU score at beam size 5 while vMF sees an improvement of only 0.8. As the original paper gives competitive results when compared to softmax models based on greedy search, we make beam search more effective for the vMF-based model a prime area for improvement.

5 Proposed Approach

The issue with beam search primarily stems from the low quality of nearest neighbors in the output embedding space in terms of decoding. This is directly demonstrated by the lack-luster growth in top-k accuracy compared softmax models shown in Table 2. We intend to explore options to better learn this output space. Kumar and Tsvetkov [2019] report that the naïve solution to this problem, allowing the model to learn the output space parameters, leads to slower optimization and worse performance.

Thus, our potential solution entail either reshaping the embedding space during learning in a more sophisticated way or learning to better interpret the space at decoding time. Approaches which reshape the embedding space would be related to normalizing flows insofar as the model is trying to learn mapping from a simpler distribution to a more complex one (such as the embedding space). Improving the search process itself could include incorporating a language model-based reranker for nearest neighbor search in order to prune away the lowest quality beam search candidates. In both approaches, elements of the decoding objective will be incorporated into the model training as the original

Contextual word embeddings are another possible avenue for improvement. Our experiments highlight that the quality of the embedding space of the target language is a bottleneck of the model. We hypothesize that the space of contextual word representations, such as BERT [Devlin et al., 2019], might be better suited than the space of traditional word vectors like Glove [Pennington et al., 2014] or FastText [Bojanowski et al., 2016]. The use of contextual word embeddings is trivial at training time since the target sentence is available. A first approach would involve multi-task training of the continuous generation model to predict both the contextual and the static word embeddings, while only using the static word embeddings at inference time. We believe this would provide improved training for the model.

6 Conclusion

Directly generating embeddings as the output of an NMT decoder is an alternative to the typical softmax output layer. The primary benefits of such a model include fewer parameters, faster training, and the ability to handle larger vocabularies.

In addition to these points, this approach offers better generalization to rare words, and output in a meaningful metric space, which can allow for backpropagation directly through the model's output as well as other techniques associated with continuous embedding spaces. The major drawback of this output paradigm is the inability to effectively decode sentences beyond a simple greedy search (e.g., decoding via beam search). As such, we propose to improve this model and allowing more effective search through sentence decoding. Potential improvements include using contextual embeddings, dynamically reshaping the output space, and using more sophisticated search procedures. We believe improving decoding for continuous output models could present a compelling alternative to traditional softmax-based decoding models.

A Appendix

Hyper parameter	Softmax	vMF
Loss	Cross entropy	vMF ($\lambda_1 = 0.02$, $\lambda_2 = 0.1$)
Encoder layers	1	1
Decoder layers	2	2
Hidden size	1024	1024
Source embedding size	512	512
Target embedding size	300	300
Embedding type	Fasttext	Fasttext
Min Word Frequency	1	1
Max Sentence Length	100	100
Source Vocabulary	50000	50000
Target Vocabulary	50000	50000
Learning rate	0.0002	0.0005
Optimizer	Adam	Adam
Input Feeding	True	True
Weight Tying	True	NA
Unk replacement	True	True

Table 3: Hyperparameters for both baseline models

Error	Ground truth	Example
Repeating n-grams	How to do it?	How do this do this? Do this do this? Do it do this? Do it do this? Do it do this? Do it do this? Do it do this?
Unrelated translation	Grit is having stamina.	Smarts is superorganism.
Similar words, different content	Now, like Doc Edgerton, a scientist himself, science became art, an art of ultra-fast photography, and I realized that all the gigabytes of data that we're collecting every time is not just for scientific imaging, but we can also do a new form of computational photography with time-lapse and color-coding, and we look at those ripples. Remember, the time between each of those ripples is only a few <unk> of a second.	So, like Leonard Brockington, even a scientist, is it become an art, and is the art of impressionistic photography, and I realized that all this granularity of data that we collect every time, we can also use a new form of computer photography, with the so-and-a-half linearities, and we can look at these waves. Don't forget, the time between every these wave is just a few 4.5 seconds.

Table 4: Examples of common errors with the vMF model

References

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *CoRR*, abs/1607.04606, 2016. URL http://arxiv.org/abs/1607.04606.

O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, V. Logacheva, C. Monz, et al. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198, 2016.

M. Cettolo, N. Jan, S. Sebastian, L. Bentivogli, R. Cattoni, and M. Federico. The iwslt 2016 evaluation campaign. In *International Workshop on Spoken Language Translation*, 2016.

W. Chen, D. Grangier, and M. Auli. Strategies for training large vocabulary neural language models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016. doi: 10.18653/v1/p16-1186. URL https://doi.org/10.18653/v1/p16-1186.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

S. Kumar and Y. Tsvetkov. Von mises-fisher loss for training sequence to sequence models with continuous outputs. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=rJlDnoA5Y7.

M. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. URL http://arxiv.org/abs/1508.04025.

F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, AISTATS 2005, Bridgetown, Barbados, January 6-8, 2005*, 2005. URL http://www.gatsby.ucl.ac.uk/aistats/fullpapers/208.pdf.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

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

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016. doi: 10.18653/v1/p16-1162. URL https://doi.org/10.18653/v1/p16-1162.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.