Error-Correcting Codes For Approximate Neural Sequence Prediction

Anonymous ACL submission

Abstract

We propose a novel neural sequence prediction method based on *error-correcting codes* that avoids exact softmax normalization and allows for a tradeoff between speed and performance. Error-correcting codes represent predictions and targets as a binary code where each bit is represented by a logit. The codebook is arranged such that similar tokens are close to each other using word embedding similarity, ensuring that incorrect predictions are at least semantically close to the target. We also address the well-established problem of compounding errors by mixing the latent codes of past predictions and past targets in one of two ways: (1) according to a predefined sampling schedule or (2) a differentiable sampling procedure that replaces the argmax operation. Low dimensional codes show similar performance to models that use the full softmax and outperform alternative approximate methods for language modeling and text generation, while generation further benefits from our mixture sampling.

1 Introduction

Unconditional and conditional language modeling (CLM) are fundamental tasks that underlie various tasks in natural language processing (Sundermeyer et al., 2012; Ghosh et al., 2016; Vaswani et al., 2017; Devlin et al., 2018). The goal is to learn a joint probability distribution for a sequence of length T containing words from a vocabulary $\mathcal{V}$ where joint distribution can be decomposed into the conditional distributions of current tokens given past tokens using the chain rule as $P(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(w_t|w_{t-1}, \ldots, w_1)$. A Recurrent Neural Network (RNN) $f_\theta(\cdot)$, parameterized by θ, can be used to encode the information at each timestep t into the last L-th hidden state vector $\boldsymbol{h}_t^L$ which is followed by a decoder $g_\phi(\boldsymbol{h}_t^L)$ which outputs a probability distribution $\hat{p}_\theta(y_t|x_t, h_{t-1})$. However, (1) training autoregressive models can be slow when $|\mathcal{V}|$ is large, while also leaving a large memory footprint for the respective input and output layers; and (2) sequence predictors suffer from *exposure bias* (EB), which refers to the compounding errors at test time due to the discrepancy between train and test time behavior i.e model is trained with maximum likelihood and assumes inputs are i.i.d, whereas at test time the model depends on previous predictions as input. Error-correcting codes (Hamming, 1950) address the two aforementioned challenges by (1) having the flexibility to trade-off between output dimensionality and performance via the code length and allocated error-checks (e.g Hierarchical Softmax (Morin and Bengio, 2005) does not allocate more dimensions for difficult to predict tokens) and (2) the latent error codes enable us to mix discrete latent factors between predictions and targets that can improve the mitigation of exposure bias (such granularity in the mixing process is not possible with current methods such as Scheduled Sampling (Bengio et al., 2015) and variants thereof (Goyal et al., 2017)). Hence, we propose an error-correcting output code (ECOC) based Neural Sequence Prediction (ECOC-NSP) model that addresses the two aforementioned challenges. We show that when given sufficient error codes ($|\mathcal{V}| \gg |c| \gg \log_2(|\mathcal{V}|)$), while the codeword dimensionality $|c| < |\mathcal{V}|$, accuracy is close to the full softmax (SM). Additionally, we create well-separated codes by rank ordering the codebook using pretrained embedding similarity where the number of error-correcting codes assigned to a token in the codebook is proportional to the cosine similarity between the tokens corresponding pretrained word embedding and the most frequent tokens word embedding. Lastly, ECOC-NSP can be improved for CLM by mitigating compounding errors using our proposed *Latent Variable Mixture Sampling* (LVMS). ECOC-NSP with LVMS outperforms the Hierarchical Softmax-based NSP that uses Scheduled Sampling (Bengio et al., 2015) and other related baselines.

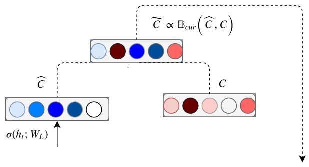

Figure 1: Curriculum Mixture Sampling

2 Methodology

A challenging aspect of assigning codewords is ordering the codes such that even if incorrect predictions are made, that the codeword is at least semantically closer to that of the codewords that are less related, while ensuring good separation between codes. Additionally, we have to consider the amount of error-checking bits to use. In theory, $\log_2(k)/k$ is sufficient to account for all k classes. However, lower bit codes can bottleneck the decoder and lack expressivity when modeling the dependencies between the output distribution. Hence, we also consider a large amount of error-checking bits. The most naive way to create the codebook is to assign binary codes to each word in random order. However, it is preferable to order codes corresponding to tokens $w \in \mathcal{V}$ proportional to their similarity while maximizing the separability between codewords that are more likely to be incorrectly predicted. Apart from this *row separability* requirement, we must choose the dimensionality of C e.g $|\log_2(\mathcal{V})| \leq d \leq |\mathcal{V}|$ bits to represent all classes with the remaining error-checking bits. We propose to reorder $C \in \mathcal{C}$ such that the Hamming distance between any two codewords is proportional to the embedding similarity and thus assigning the amount of error-checking bits for a given token proportional to the rank ordered similarity for a chosen query word embedding. In our experiments we use pretrained GoogleNews `skipgram` embeddings.[1] Words with high similarity have codes that have lower Hamming distance $\mathrm{H}(\cdot,\cdot)$. This ensures that even when codes are correlated, incorrect latent predictions are semantically closer to the targets.

2.1 Latent Variable Mixture Sampling

To mitigate EB for latent code models we propose a sampling strategy that interpolates between predicted and target codewords. We refer to this as Latent Variable Mixture Sampling (LVMS) and its application to ECOC as *Codeword Mixture Sampling* (CMS). In Curriculum-Based Latent Variable Mixture Sampling (CLVMS), the mixture probability is $p_c = 0\ \forall c \in C$ at epoch $\epsilon = 0$ and throughout training the probability monotonically increases $p_c = \delta_c\ \forall c \in C$, where δ_c is the threshold for the c-th bit after ϵ epochs. A Bernoulli sample $\tilde{C} = \mathbb{B}(\hat{C}_c, C_c)\ \forall c \in [0, C]$ is carried out for $t \in T$ in each minibatch. The probabilities per dimension p_c are independent of keeping a prediction $\hat{y}_{t-1,c}$ instead of the c-th bit in the target codeword $y_{t-1,c}$ at timestep t-1. The reason for having individual mixture probabilities per bits is because when we consider a default order in $\mathcal{C}$, this results in tokens being assigned codewords ranked by frequency. Therefore, the leftmost bit predictions are more significant than bit errors near the beginning (e.g $2^0 = 1$ only 1 bit difference). We report results for a sigmoidal schedule as shown in Equation 1 where τ_{max} represents the temperature at the last epoch, δ is a scaling factor controlling the slope and $\forall \epsilon \in [-\mathrm{N}/2, \mathrm{N}/2]$.

$$[\hat{y}_{t-1}, y_{t-1}] \sim \tau_{max} / \big(1 + \exp(-\epsilon/\delta)\big) \qquad (1)$$

Unlike scheduled sampling, we can sample a mixture of the predicted and target factored distributions that represents the posterior (i.e not only prediction or target but a mix of their latent codes). This is illustrated in Figure 1 where the color strength illustrates the activation between $[0, 1]$.

Latent Soft-Mixture Sampling In standard CMS, we pass the token index w_t, which is converted to an input embedding e_w based on the most probable bit predictions at the last time step, $\mathrm{argmax}_\theta\, p(y_{t-1}|x_{t-1}; \theta)$. We can instead replace the argmax operator with a soft argmax that uses a weighted average of embeddings $e \in E$ where weights are assigned from the previous predicted output via the softmax normalization $\phi(x_{t-1}, \tau)$, where τ controls the kurtosis of the probability distribution ($\tau \to 0$ tends to argmax) in Equation 2.

$$x_t = \sum_{w \in \mathcal{V}} e_w \left(\frac{\exp(h_w^T \theta / \tau)}{\sum_{w \in \mathcal{V}} \exp(h_w^T \theta / \tau)} \right) \qquad (2)$$

In the ECOC-NSP, we consider binary codewords and therefore choose the top k least probable bits to flip according to the curriculum schedule. Hence, this results in k codewords where each $\hat{C}$ has at least Hamming distance $\mathrm{H}(\hat{C}, C) = 1\ (2^0)$. Concretely, this is a soft interpolation between past targets and a weighted sum of the k most probable codewords $\hat{C}_K = \mathrm{argmax}_k \left(\sigma(h_w^T W)\right)$ such that $x_t = \mathbb{B}_K\Big(C, \sum_k^K \phi(\hat{C}_k)\Big)$ where B_K samples one or the other for each kth dimension of C.

[1] see here: https://code.google.com/archive/p/word2vec/

2.2 Differentiable Latent Variable Sampling

To directly differentiate through the origin of cascading errors (unlike scheduled sampling), we extend the use of differentiable scheduled sampling (Goyal et al., 2017) to mixture sampling by replacing the argmax operation with the Concrete distribution (Maddison et al., 2016) to adjust gradients where prior predictions changed value throughout training. This not only identifies at which time-step the error occurs, but what latent variables (i.e. output codes) had the most influence in generating the error. We sample latent codes inversely proportional to the errors from a Gumbel distribution, as this distribution has shown to resemble the errors of logistic regression models, similar to the logits corresponding to each bit in the code. Similarly, instead of passing the most likely predicted word $\hat{y}^{w*}_{t-1}$, we can sample from $\hat{y}_{t-1} \sim \phi(h_{t-1}, w)$ and then pass this index as $\hat{x}_t$. This is an alternative to always acting greedily and allow the model to seek other likely actions. However, to compute derivatives through samples from the softmax, we need to avoid discontinuities such as the argmax operation. The Gumbel-Softmax (Maddison et al., 2016; Jang et al., 2016) allows us to sample and differentiate through the softmax by providing a continuous relaxation that results in probabilities instead of a step function (i.e. argmax). As shown in Equation 3, for each componentwise Gumbel noise $k \in [1.., n]$ for latent variable given by $h^T\theta$, we find k that maximizes $\log \alpha_k - \log(-\log U_k)$ and then set $D_k = 1$ and $D \neg k = 0$, where $U_k \sim \text{Uniform}(0, 1)$ and α_k is drawn from a discrete distribution $D \sim \text{Discrete}(\alpha)$.

$$\hat{p}(y_t|x_t; \theta) = \frac{\exp((\log \alpha_k + G_k)/\tau)}{\sum_{i=1}^{n} \exp((\log \alpha_i + G_i)/\tau)} \quad (3)$$

For ECOC, we instead consider Bernoulli random variables for which the Concrete distribution can be expressed by means of two arbitrary Gumbel distributions G_1 and G_2. Sampling a Binary Concrete random variable involves sampling Z, sample $L \sim$ Logistic and set Z as shown in Equation 4, where $\alpha, \tau \in (0, \infty)$ and $Z \in (0, 1)$.

$$Z \equiv 1/\big(1 + \exp(-(\log \alpha + L)/\tau)\big) \quad (4)$$

This is used for ECOC and other latent variable-based models, such as Hierarchical Softmax (HS; Mnih and Hinton, 2009), to propagate through past decisions and make corrective updates that backpropagate to where errors originated from along the sequence. Hence, we also carry out experiments with BinConcrete (Equation 4) and Gumbel-Softmax(Equation 3) for HS and ECOC respectively. In this work, we consider using an annealed τ, similar to Equation 1 where $\tau \to 2.5$ and starts with $\tau = 0.01$. This allows the model to avoid large gradient variance early in training. For the Gumbel-Softmax in LVMS, this corresponds to the model becoming more robust to non-greedy actions gradually throughout training.

Experimental Details. Experiments are carried out for a 2-hidden layer Long-Short Term Memory (LSTM) model with embedding size $|e| = 400$, Backpropogation Through Time (BPTT) length 35 and variational dropout (Gal and Ghahramani, 2016) with rate $p_d = 0.2$ for input, hidden and output layers. The ECOC-NSP model is trained using the loss shown in Equation 5, where k is a group of error-checking codewords corresponding to a codeword C and $\hat{y} = \sigma_c(\boldsymbol{h}^\top\theta)$.

$$\mathcal{L}_\theta = \max_k \prod_c^C \big[y_c \log \hat{y}_c + (1\text{-}y_c) \log(1 - \hat{y}_c))\big] \quad (5)$$

The gradients can then be expressed as $\frac{\delta \mathcal{L}}{\delta \theta} = (y - \sigma(\boldsymbol{h}^\top\theta))\boldsymbol{h}^\top$. For prediction, we then choose the most probable code (some of which may be error-checks) and predict its corresponding token. We first compare our proposed ECOC-NSP to methods that approximate softmax normalization, using binary trees and latent codes that are ordered according to unigram frequency (Uni-Hierarchical-SM and Uni-ECOC). These baselines are the Sample-Softmax (Bengio et al., 2003; Bengio and Senécal, 2008), HS, AS (Grave et al., 2016) and NCE (Mnih and Teh, 2012)) to our ECOC-NSP approach. For text generation, we also include SS and soft-SS with SM (Soft-SS-SM) as the baselines, to compare against the proposed mixture sampling techniques.

3 Results

Language Modeling Results. Table 1 shows that overall ECOC with a rank ordered embedding similarity $\mathcal{C}$ (Embedding-ECOC) **almost performs as well as the full-softmax (8.02M parameters) while only using 1000 bits for PTB** ($|\mathcal{V}|/20$ and) and 5K bits for WikiText-2 ($|\mathcal{V}|/25$) and WikiText-103 ($|\mathcal{V}|/30$). The HS-based models use a 2-hidden layer tree with 10 tokens per class, resulting in 4.4M parameters for PTB, 22.05M parameters for WikiText-2 (full softmax - 40.1M) and WikiText-103. Moreover, we find there is a **consistent im-**

Model	PTB Val.	PTB Test	WikiText-2 Val.	WikiText-2 Test	WikiText-103 Val.	WikiText-103 Test
Full SM Gal and Ghahramani	***86.19***	***79.24***	***124.01***	***119.30***	***56.72***	***49.35***
Rand-Sample-SM Bengio and Senécal	92.14	81.82	136.47	129.29	68.95	59.34
Uni-Sample-SM Bengio and Senécal	90.37	81.36	133.08	127.19	66.23	57.09
Rand-Hierarchical-SM Morin and Bengio	94.31	88.50	133.69	127.12	62.29	54.28
Uni-Hierarchical-SM Morin and Bengio	92.38	86.70	130.26	124.83	62.02	54.11
Adaptive-SM Grave et al.	91.38	85.29	118.89	120.92	60.27	52.63
NCE Mnih and Teh	96.79	89.30	131.20	126.82	61.11	54.52
Random-ECOC	91.00	87.19	131.01	123.29	56.12	52.43
Uni-ECOC	86.44	82.29	129.76	120.51	***52.71***	***48.37***
Embedding-ECOC	***84.40***	***77.53***	***125.06***	***120.34***	57.37	49.09

Table 1: **LSTM Language Modeling Test Perplexities**.

provement in using Embedding-ECOC over using a random codebook (Random-ECOC) and a **slight improvement over using a unigram ordered codebook** (Uni-ECOC). Note that in both Embedding-ECOC and Uni-ECOC, the number of error-checking bits are assigned inversely proportional to the rank position when ordering embedding similarities and unigram frequency respectively. We also found that too many bits (e.g. $|C| = |\mathcal{V}|$) take much longer ($\epsilon \in$ [20-30] more for PTB) to converge with negligible perplexity reductions. Hence, the advantage of ECOC-NLVMS is the large compression rate while maintaining performance e.g when using a codebook dimensionality of $|C| = 40$ for PTB, we observe test perplexity that is within 2 perplexity points for the same model that uses the full softmax.

Code Length vs Performance. Figure 2 shows the reduction in perplexity with the increase in bits in ECOC-LSTM decoder parameters. For PTB, large perplexity reductions are made between 14-100 codebits, while between 100-1000 codebits there is a gradual decrease. In contrast, we see that there is more gained from increasing codeword size for WikiText-2 and WikiText-103 (which preserve the words that fall within the long-tail of the unigram distribution). Intuitively, increasing code length and error-checks reduces test perplexity.

Latent Variable Mixture Sampling Text Generation Results. Table 2 shows all results of LVMS when used in HS and ECOC-based NSP models for the MSCOCO image captioning dataset (Lin et al., 2014) with $|c| = 200$ to account for vocabulary size $|V| = 10^3$, leaving $|c| - \log_2(|V|) = 186$ error-check bits leftover $\forall C \in \mathcal{C}$. The HS uses the Categorical Concrete distribution for DLVMS-HS and Binary Concrete Distribution for DCMS-ECOC. Both HS and ECOC use an Embedding ordered decoder matrix (we omit the -Embedding extension). This is baselined against both SS and the soft-argmax version of SS, the most related sample-based supervised learning approach to LVMS. Additionally, we report results on CLVMS-ECOC (Curriculum-LVMS ECOC) that mixes prediction and target codewords using the schedule in Equation 1 and a differentiable extension of LVMS via samples from the Gumbel-Softmax (DCMS-ECOC). DCMS-ECOC and DLVMS-Hierarchical-SM both sample from each softmax along the tree branch to the target code at training time. We find that using a curriculum in CLVMS-ECOC with a **semantically ordered codebook outperforms the full softmax with scheduled sampling** (SS-SM) and its weighted-variant (Soft-SS-SM). Moreover, DLVMS-ECOC further improves over CLVMS-ECOC on MSCOCO and **LVMS make a consistent improvement over SS, suggesting LVMS is an effective NSP alternative**.

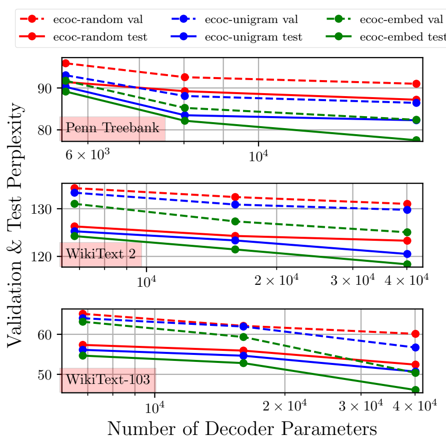

Figure 2: **ECOC-NSP Perplexity vs. Decoder Parameters** (corresponding to 14/20/40 codeword bits for Penn-TreeBank and 17/40/100 codeword bits for WikiText-2/103)

	B1	B2	B3	B4	R-L	MET
Full-SM Gal and Ghahramani	71.09	51.33	32.85	24.67	50.28	52.70
SS-SM Bengio et al.	73.23	52.81	33.37	26.11	52.60	54.51
Soft-SS-SM Goyal et al.	***73.54***	***53.01***	***33.26***	***27.13***	***54.49***	***54.83***
SS-Adaptive-SM Grave et al.	70.45	50.22	31.38	23.59	51.88	51.83
SS-Hierarchical-SM	67.89	48.42	30.37	22.91	49.39	50.48
CLVMS-Hierarchical-SM	69.70	49.52	31.91	24.19	51.35	51.20
DLVMS-Hierarchical-SM	71.04	50.61	32.26	24.72	52.83	52.36
SS-ECOC	72.02	52.03	32.57	25.42	51.39	53.51
Soft-SS-ECOC	72.78	53.29	33.15	25.93	52.07	54.22
CLVMS-ECOC	***74.70***	***53.09***	***34.28***	***27.05***	***53.67***	***55.62***
DLVMS-ECOC	***74.92***	***53.56***	***34.70***	***27.81***	***54.02***	***55.85***

Table 2: **MSCOCO Test Results** on BLEU (B), ROUGE-L (R-L) & METEOR (MET) Evaluation Metrics.

4 Conclusion

We proposed an error-correcting neural language model to approximate the softmax and a novel Latent Variable Mixture Sampling method to mitigate exposure bias. Performance is maintained close to models that use the full softmax and related approximate methods with drastically lower code lengths. Lastly, mixture sampling and its differentiable variants are complementary to error-correcting codes and effectively mitigate exposure bias. In future work, we extend error codes to Transformers.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713–722.

Yoshua Bengio, Jean-Sébastien Senécal, et al. 2003. Quick training of probabilistic neural nets by importance sampling. In *AISTATS*, pages 1–9.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual lstm (clstm) models for large scale nlp tasks. *arXiv preprint arXiv:1602.06291*.

Kartik Goyal, Chris Dyer, and Taylor Berg-Kirkpatrick. 2017. Differentiable scheduled sampling for credit assignment. *arXiv preprint arXiv:1704.06970*.

Edouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. 2016. Efficient softmax approximation for gpus. *arXiv preprint arXiv:1609.04309*.

Richard W Hamming. 1950. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.