BrainQA: Discrete Representations for Question Answering

Joseph Redling-Pace
jjr263

Jonathan Friedman
jif24

Abstract

Humans are able to summarize and organize the world into discrete concepts. This ability allows us to efficiently store and recall memories. In the realm of question-answering, machine learning has struggled with performance on longer passages, such as non-extractive question answering on whole books. Inspired by the role of discrete memory representations in the mammalian hippocampus, we propose an end-to-end question answering model utilizing discrete representations for high-level concept learning. Testing this model on the perhaps "easier" task of extractive question-answering, we find that a discrete bottleneck actually hinders performance, even accounting for latent space issues such as index-collapse.

1 Introduction

Humans make use of discrete concepts to both represent the world around us and structure memory. For example, someone can summarize a conversation by saying "we talked about X and Y", without needing to recall every sentence. Deep inside the cortex of the brain, certain cells in the mammalian hippocampus have been found to represent discrete events (Sun et al., 2020). Computational models of the hippocampus have shown how clustering algorithms can actually account for some of this behavior, replicating unique patterns of hippocampal neural activity. (Mok and Love, 2019).

Machine reading comprehension (MCR) is a question answering task where a model is fed a passage and then asked questions relating to that passage. Although state-of-the-art models (typically ensemble based) exceed human performance on datasets such as SQuAD2.0, as the passages get longer typical RNN and LSTM based approaches start to fail (Rajpurkar et al., 2018; Rae et al., 2019)

Our model proposes a discrete bottleneck for extracting high level themes. We argue that by training end-to-end, our model is forced to learn more general semantic concepts, thereby improving its performance downstream. We make use of a Vector Quantized Variational AutoEncoder (VQ-VAE) as a bottleneck in conjunction with a pre-trained BERT model for question answering (van den Oord et al., 2017; Wolf et al., 2019). [1]

2 Related Work

A primary challenge in working with discrete values in machine learning is that backpropagation requires continuous values to compute gradients. Active research in this area has led to several solutions to this problem. The Gumbel-Softmax reparameterization trick, for example, enables differentiable samples to be taken from a discrete latent space (Jang et al., 2016).

One popular model for learning discrete latent spaces is the VQ-VAE, which was later improved upon through use of a hierarchically structured latent space in VQ-VAE-2 (van den Oord et al., 2017; Razavi et al., 2019). VQ-VAE and VQ-VAE-2 were implemented in the context of generating high-fidelity images and reconstructing human speech, however they have since been used for additional tasks such as machine translation, and recently in OpenAI's Jukebox - a music generating system (Łukasz Kaiser et al., 2018; Dhariwal et al., 2020). In general, VAEs have struggled to match the performance of typical recurrent models in NLP (Bowman et al., 2016). However, additional research has shown that they can in fact achieve competitive performance (Yang et al., 2017).

3 VQ-VAE

VQ-VAE shares a similar encoder-decoder architecture as a typical VAE, however it has an additional step to quantize its latent space. Instead of using the Gumbel-Softmax reparameterization trick, VQ-

[1] All code can be found on: https://www/github.com/yifr/brainqa

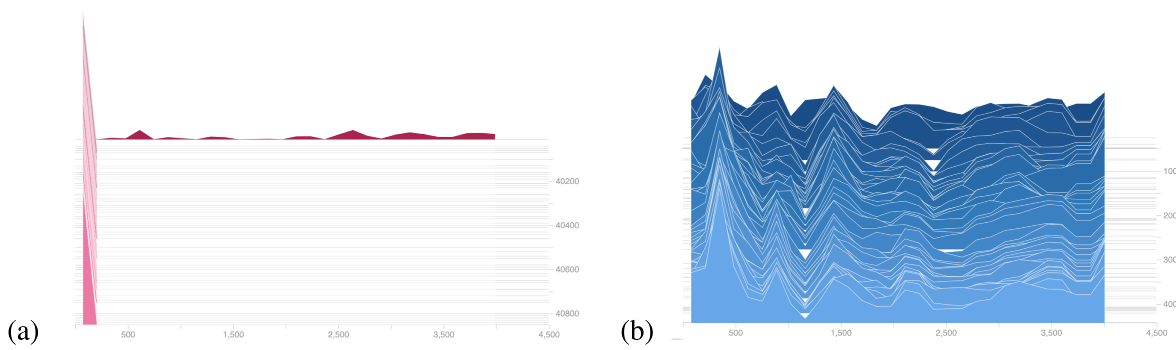

Figure 1: Solving index collapse. These histograms show usage of discrete latent codebook indices over time. (a) Index collapse using vanilla VQ-VAE. Early on in training, VQ-VAE collapses to using just a few latent indices and remains that way throughout training. (b) VQ-VAE wth random restart - although VQ-VAE prefers the same latent spaces, they now have a much more even distribution.

VAE approximates a loss function as follows. The encoder output $z_e(x)$ is passed through a discretization bottleneck using a nearest-neighbor lookup on embedding vectors $e \in \mathbb{R}^{K \times D}$. This maps each sequence in our encoder output to 1-of-K discrete latent spaces, giving us a "codebook" of discrete concepts. With E as our encoder:

$$Quantize(E(x)) = e_k \tag{1}$$

$$e_k = \arg\min_{j \in [K]} \|E(x) - e_j\|_2 \tag{2}$$

Our decoder input is therefore the discrete latent vector closest to the encoder output, as measured in ℓ_2 distance. The VQ-VAE loss is defined by three terms. The first term is the reconstruction loss $\log p(x|z_q(x))$, defined by the ℓ_2 error between the encoder input and the decoder output. The second term moves the embeddings in the codebook closer to the outputs of the encoder. The third term is called a "commitment loss", which motivates the encoder to stick to an embedding representation. If $D(x)$ is the reconstruction output from our decoder, the total loss function for VQ-VAE is as follows:

$$\mathcal{L} = \mathcal{L}_{recon} + \mathcal{L}_{codebook} + \beta \mathcal{L}_{commit} \tag{3}$$

$$\mathcal{L}_{recon} = \frac{1}{T} \sum_t \|x_t - D(x)\|_2^2 \tag{4}$$

$$\mathcal{L}_{codebook} = \|\text{sg}[z_e(x)] - e_j]\|_2^2 \tag{5}$$

$$\mathcal{L}_{commit} = \|z_e(x) - \text{sg}[e_j]\|_2^2 \tag{6}$$

Where sg$[\cdot]$ is the stopgradient operator, defined as the identity function during the forward computation and has zero partial derivatives in the backward computation.

In that it makes use of clustering mechanisms to construct discrete latent representations, VQ-VAE is suitable for the purpose of approximating our desired cognitive mechanisms. However, it has several limitations, two of which we address here.

3.1 Index Collapse

Vanilla VQ-VAE is known to suffer a problem referred to as *index collapse*, in which the latent space collapses to using just a few of the many codebook indices. We solve this problem using the random restart method from (Razavi et al., 2019). With this approach, anytime a particular codebook index falls below a mean usage of a specific threshold, we randomly assign it to an encoder output in the current batch. This ensures that all codebook indices are used, preventing index collapse.

It is worth noting that, although it is the strategy we adopt, random restart is not the only technique that has been proposed for solving index collapse. (Łukasz Kaiser et al., 2018) details two other strategies for addressing this issue - one in which encoder outputs are split into n_d smaller slices before quantization, and one in which the encoder is given a fixed set of projections at the start of training.

As we will discuss later on, we found that for our specific use case, solving index collapse did not improve results. One possible reason why is that random restart is an unnatural solution for our dataset. SQuAD2.0 features a number of questions for each passage, so by forcing two similar passages to map to opposite ends of the discrete latent space, we may be "confusing" our VQ-VAE more than helping it. In the future, it would be worthwhile to experiment more with different strategies for solving this problem.

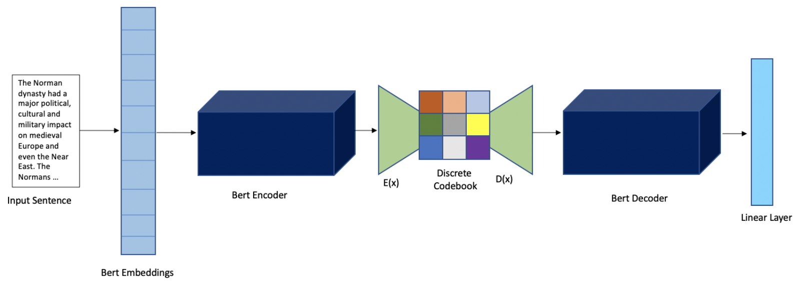
The Norman dynasty had a major political, cultural and military impact on medieval Europe and even the Near East. The Normans ...
Input Sentence
Bert Embeddings
Bert Encoder
E(x)
Discrete Codebook
D(x)
Bert Decoder
Linear Layer

Figure 2: BrainQA Model Architecture

3.2 Sampling from the Latent Space

An additional challenge that comes with the discrete latent space in VQ-VAE is how to draw samples from our prior $p(z)$. Because we cluster representations, we cannot sample uniformly from z, as we do not have meaningful representations distributed uniformly. In order to learn a prior $p(z)$, the authors of VQ-VAE suggest training an additional model over the latent space once training is completed to learn an autoregressive distribution. Our initial plan was to do online sampling, and reconstruct semantically similar BERT representations from our VQ-VAE as a form of memory. However, the secondary model constraint made that architecture challenging. Instead we defaulted to our current architecture, in which VQ-VAE plays a more auxiliary role as a "concept-bottleneck," which will be described in greater detail in the coming section.

4 BrainQA

Our model makes use of pre-trained BERT models for language processing and question answering. We feed our encodings through a pre-trained BERT encoder. The hidden state output of the encoder is fed to VQ-VAE, which acts as a bottleneck to extract high level semantic concepts. We subsequently feed the reconstructed hidden states to our BERT decoder. The output of our BERT decoder is fed into a linear layer which outputs predictions for question-answer spans. The VQ-VAE architecture makes use of its own encoder and decoder to map hidden states to a discrete latent space. Inspired by (Łukasz Kaiser et al., 2018), our encoder uses a series of convolutions followed by a residual layer to process the hidden state input. For more details please refer to figure 4. Our vector quantizer uses a k-nearest neighbor algorithm explained above, and the decoder performs the inverse operation of the encoder to reconstruct a hidden state from a latent embedding e_k. During training we add our VQ-VAE loss to our total loss in order to optimize both models end to end. In doing so, we enforce a discrete bottleneck whereby our model needs to uncover an underlying discrete representation for the passage it is processing. This discrete representation ideally corresponds to some high-level feature, such as a concept, theme, or style. The intention was to improve model performance by encouraging additional semantic processing. Instead, we found that this bottleneck had quite the opposite effect.

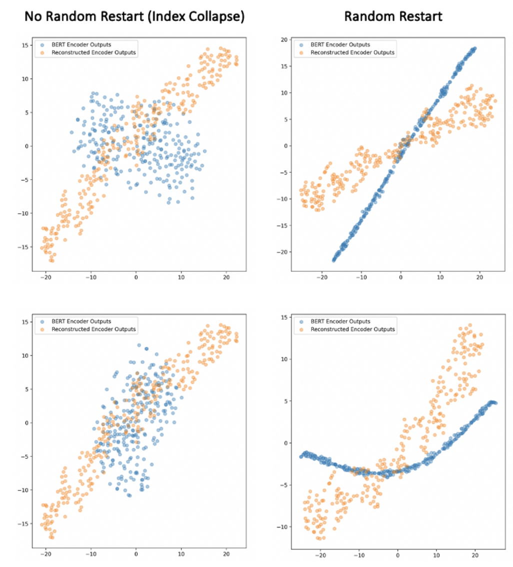

Figure 3: The column on the left shows how, when the VQ-VAE experiences index collapse, the reconstructions tend to be homogeneous, whereas a random restart policy allows them to improve, although not drastically

5 Discussion

We experimented with a number of different architectures and conducted several ablation studies to

BrainQA Results

Metric	Baseline	Vanilla-256	Vanilla-4096	RR-256	RR-4096
exact	**63.40**	*43.43*	39.46	42.83	42.21
f1	**67.18**	*44.56*	41.21	44.14	43.49
HasAns_exact	**64.15**	7.56	*9.19*	7.81	7.86
HasAns_f1	**71.74**	9.82	*12.67*	10.44	10.43
NoAns_exact	62.64	**79.21**	69.64	77.75	76.45
NoAns_f1	62.64	**79.21**	69.64	77.75	76.45

Table 1: Vanilla-K corresponds to VQ-VAE with no random restart for index collapse whereas RR-K refers to VQ-VAE with random restart. K refers to how many latent spaces contained in our VQ-VAE. Best overall scores are bolded and best of our models are italicized.

try to improve model performance and tune hyperparameters. We found that using VQ-VAE to reconstruct BERT embeddings had an overall worse performance than using VQ-VAE to reconstruct encoder hidden states. We suspect this is because aberrations in embedding reconstructions having more downstream impact than aberrations in encoder outputs. As such, we only report the tests conducted with the model architecture detailed in 2. We report results using both 256 and 4,096 latent states for VQ-VAE, and the corresponding difference when using a random restart policy. (Łukasz Kaiser et al., 2018) found that using vanilla VQ-VAE for machine translation gave a BLEU score of 2.56. Hence, the results we obtained with vanilla VQ-VAE were not particularly surprising. Note that we obtain better scores for guessing NoAns than the baseline, simply because that is the dominant strategy for a model performing worse than random guessing. What *is* rather surprising, however, is how poor the model continues to do even after correcting for index collapse.

Applying T-SNE to visualize encoder outputs and their corresponding reconstructions (See 3), we find that, although our random restart policy does in fact do a slightly better job reconstructing latents, it still falls very short of acceptable. It is possible that our joint optimization scheme does a poor job training our latent space. In the future, we can experiment with a different embedding loss policy, such as exponential moving averages (EMA).

6 Conclusions

VQ-VAE, and VAEs in general are powerful generative tools. The extractive question answering task we apply our model to here is perhaps not the best domain for such a tool. The notion of discrete latent representations seems more effective for tasks that have clear categorical applications, such as controlled audio, image or text generation.

The model presented here differed rather significantly from our initial proposal. Future work in the domain of discrete concept learning for language tasks can take several forms. Perhaps, extractive question answering is not the correct domain to utilize this tool. There are generative question answering or summarization tasks that may find greater benefit from explicit concept learning models.

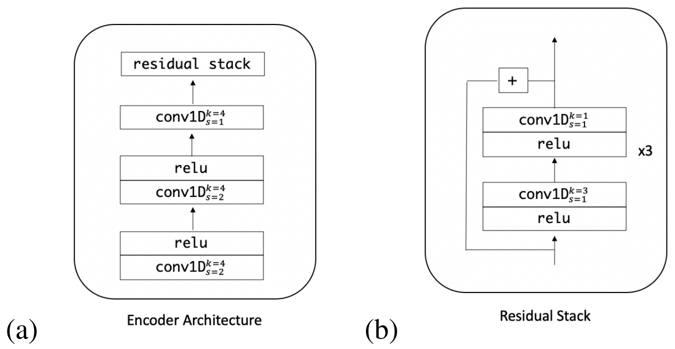

Figure 4: Architectural diagram for our VQ-VAE Encoder and residual stack. The decoder performs an inverse operation as the encoder. The superscript k indicates kernel size and the subscript s indicates stride length.

References

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*.

Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. 2020. Jukebox: A generative model for music. *arXiv preprint arXiv:[TODO]*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax.

Łukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables.

Robert M. Mok and Bradley C. Love. 2019. A non-spatial account of place and grid cells based on clustering models of concept learning. *Nature Communications*, 10(1):5685.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. 2019. Compressive transformers for long-range sequence modelling.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Ali Razavi, Aaron van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with vq-vae-2.

Chen Sun, Wannan Yang, Jared Martin, and Susumu Tonegawa. 2020. Hippocampal neurons represent events as transferable units of experience. *Nature Neuroscience*, 23(5):651–663.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions.