Two-Stream Transformer Architecture With Discrete Attention for Better Interpretrability and Separation of Model Concerns

Two-Stream Transformer Architecture with Discrete Attention for Better Interpretrability and Separation of Model Concerns

Jambay Kinley

Adviser: Professor Alexander Rush

An undegraduate thesis submitted in partial fulfillment of the requirements for the degree of Bachelor of Arts in Computer Science with Honors

Harvard University
Cambridge, Massachusetts
April 2020

Abstract

The transformer has become a central model for natural language processing tasks ranging from translation to classification to representation learning. Its success demonstrates the effectiveness of stacked attention as a replacement for recurrence for many tasks. Attention is broadly interpreted as selectively attending to different parts on an input. So, in theory, attention offers more insights into the model's internal decisions; however, in practice, when stacked, it quickly becomes nearly as fully-connected, making it hard to disentangle final decision dependencies.

In this work, we propose an alternative transformer architecture, discrete transformer, with the goal of improving model interpretability. We use discrete latent-variable attention to ensure that decision steps only depend on a limited context. We separate out attention decisions from representation modeling by using a separate stream for each.

Empirically, on both classification and translation tasks, this approach maintains similar levels of performance on several datasets as the standard transformer, while obtaining quantitatively better attention interpretability and separating out syntactic features in the learned representations.

Finally, our two-stream formulation can be used to transfer knowledge in a multiview arithmetic evaluation task.

Acknowledgements

I would like to convey my deepest gratitude to my adviser Professor Sasha Rush for your generous mentorship and guidance. I have learned the most through my coursework and research with you. I am always inspired by your brilliance and motivated by your investment in the growth and success of all of your students.

I would also like to thank Yuntian Deng for your mentorship throughout this project. Working with you has not only taught me technical stuff but also the ways of doing research. I am confident that these lessons will serve me well in my future endeavors.

I am incredibly grateful to my amazing friends Abhishek Anand and Mirac Suzgun, who have supported me from the inception of the project to the completion of this thesis. Talking with you about the project at its various stages helped me a great deal. I am even more grateful to you both for being there for me throughout my time in college.

I want to thank my friends Abdul, Arjun, Dong, Elizabeth, Jacqie, Rohan, Sabrina, and Spencer, for believing in me and encouraging me throughout this process.

I would like to thank the Office of Undergraduate Research and Fellowship for providing me with funding that allowed me to do research work during the summer. I am immensely grateful to Harvard for providing me with all of the resources and opportunities I have had throughout my college career. I am fortunate to have spent my last four years at this incredible place of learning among amazing people.

I express my heartfelt gratitude to my parents Penjore and Gyem, my sisters Azhim and Wama, and my best friends Sangay and Ugyen for their lifelong love, support, and encouragement in all aspects of my studies. Thank you to Azhim and Acho Sonam for providing me with a home away from home. I would like to thank everyone else, including all my family, friends, and teachers, who have led me to where I am.

Finally, thank you to you, reader, for making this possible. Thank you for your time and commitment to education.

Contents

1 Introduction **5**

1.1 Thesis Structure 6

2 Related Work **7**

3 Background **9**

3.1 Attention 9

3.2 Transformer 11

3.3 Gumbel Softmax Estimator 15

4 Proposed Model Architecture **16**

4.1 Discrete Attention Transformer 16

4.2 Controller / Model Separation 17

5 Experimental Setup **20**

5.1 Tasks 20

5.2 Data 20

5.3 Model Architectures and Settings 22

5.4 Metrics 23

6 Results and Analysis **24**

6.1 Text Classification 24

6.2 Machine Translation 25

6.3 Multiview Arithmetic Evaluation 28

7 Conclusion **29**

7.1 Future Work 29

References **31**

Appendix **35**

1 | Introduction

The transformer has become a central model for natural language processing. State-of-the-art models in a wide range of sequence modeling tasks are built upon the transformer architecture. These include language modeling using GPT-2 (Radford et al., 2019), machine translation using the original transformer (Vaswani et al., 2017), question answering using GPT-2 and BERT (Radford et al., 2018; Devlin et al., 2018), among others.

Unlike recurrent neural networks, the transformer requires no recurrence and instead uses an attention mechanism to draw global dependencies between input and output. It builds up feed-forward hidden states by attending to the source side (inter-attention) and attending to its past predictions (self-attention) with multiple heads in multiple layers (Vaswani et al., 2017). Compared to recurrent models, this attention mechanism has the potential to increase decision interpretability through its softmax bottleneck (Bahdanau et al., 2014; Xu et al., 2015; Chan et al., 2015).

However, all elements in a soft attention mechanism (Luong et al., 2015) have non-zero weight, and these weights are propagated through a highly non-linear model. It is unclear whether the magnitude of attention weights inherently reflects the relative importance of the corresponding inputs (Jain and Wallace, 2019). Having high attention weight for a particular component doesn't necessarily mean that the component is more important than the other components with lower weights, making it challenging to interpret soft attention. This problem is compounded in the transformer because the multiple stacked attention layers make it harder to discriminate the contributions of each input to the final decisions made by the model.

Can we force the transformer to more cleanly separate out its routing decisions? In this work, we consider a variant of the transformer architecture with the goal of maintaining performance while forcing discrete and segmented decisions. Specifically, we consider a *discrete transformer* with two changes to the architecture: (a) attention is a categorical latent-variable (Deng et al., 2018; Shankar et al., 2018) that makes discrete attention decisions, (b) the routing mechanism is

separated into controller and representation modeling streams.

The training mechanism for our model is a slightly modified version of standard transformer training - we use the Gumbel-Softmax approximation (Jang et al., 2016; Maddison et al., 2016), which is differentiable, to approximate discrete attention. At inference time, we only use argmax "attention". Each intermediate representation is built up based on a subset of the attended lower-layer, guaranteeing that hidden states solely depend on attended elements.

To validate this approach, we perform experiments on classification and translation. For both tasks, we achieve similar performance to standard transformer models, but much stronger attention interpretability based on the metrics by Jain and Wallace (2019). Analysis also shows how the controller and model streams separate out model concerns (the controller is concerned with attention decisions while the model stream is concerned with composing modeling representations). In particular, we observe that the controller stream utilized highly-syntactic representation for routing. Finally, we apply our two-stream approach to a multi-view arithmetic reasoning problem and find that our model can transfer knowledge from one view to another via shared a representational modeling stream.

1.1 Thesis Structure

This thesis is structured as follows:

Chapter 2 reviews related work.

Chapter 3 provides the background. We provide an overview of attention mechanism as used in NLP and describe the transformer architecture. An understanding of the transformer architecture allows us to later focus only on the modifications introduced in the discrete transformer. We then discuss the Gumbel-Softmax approximator.

Chapter 4 describes the proposed model architecture. We describe the changes we make on the original transformer architecture to discretize attention and to separate the controller and representation modeling streams.

Chapter 5 describes the experimental setup. We provide descriptions of the data and model settings used for the different tasks.

Chapter 6 provides the experimental results and analysis. We evaluate the interpretibility of the models in text classification and machine translation. We then describe the separation of syntactic from semantic properties in the two-streams. We explore this phenomenon further by analysing the results of the multiview arithmetic evaluation experiment.

Chapter 7 concludes our thesis.

2 | Related Work

Attention has been used to imply transparency into a model's prediction. This is crucially important in domains such as health care (Caruana et al., 2015; Choi et al., 2016; Rajkomar et al., 2018) but has also been used in other natural language sequence modeling tasks (Rush et al., 2015; Deng et al., 2017; Alvarez-Melis and Jaakkola, 2017). Since the soft attention mechanism assigns non-zero weights everywhere, to get interpretability, a general assumption is that larger attention weights correspond to higher importance in making a decision (Unanue et al., 2018). However, a recent work (Jain and Wallace, 2019) shows that attention magnitude does not correlate well with gradient-based measures of input elements importance (Selvaraju et al., 2017).

To get around with the difficulty of credit assignment in soft attention, researchers have proposed to use sparse attentions. Peters et al. (2018) uses sparsemax (Martins and Astudillo, 2016) to induce sparse attention structures in LSTMs. Lei et al. (2016) model attention as Bernoulli random variables and use an encoder to produce the final prediction only from the attended input elements such that the final predictions can be rationalized. To optimize the final objective, Lei et al. (2016) apply policy gradients. Our work follows the spirit of their work, but we consider multi-head multi-layer attentions in the transformer, which subjects the REINFORCE algorithm to large gradient variance. Instead, we use the Gumbel-Softmax trick (Jang et al., 2016; Maddison et al., 2016) to get parameterizable samples and reduce the gradient variance (Kingma and Welling, 2013). Our work appears most similar to the recent work of adaptively sparse transformers, which also tries to induce sparsity in transformers (Correia et al., 2019), but a notable difference is that some layers in adaptively sparse transformers are dense, such that eventually every output word still depends on all input words. In contrast, in our case, every layer uses discrete attentions.

Broadly speaking, there are two directions of work towards improving interpretability: model interpretability and prediction interpretability (Alvarez-Melis and Jaakkola, 2017). Prediction interpretability relies on an external interpreter that is both interpretable and locally consistent with the black box model being

explained, through which we can analyze the causal relationships between inputs and outputs (Ribeiro et al., 2016). In model interpretability, researchers try to build models that are commonly regarded as interpretable (Louppe, 2014; Calders et al., 2013; Doshi-Velez and Kim, 2017; Peters et al., 2018). Our approach aims to improve model interpretability by modifying the attention mechanism and objective function where we implicitly assume that more sparsity in the attention structure implies more interpretability, rather than relying on another model to analyze an existing one. While our approach does not directly lead to prediction interpretability, we can draw connections between our approach and the prediction interpretability framework of Alvarez-Melis and Jaakkola (2017) if we consider local permutations of input embeddings: for inputs not being directly or indirectly attended to at a specific prediction step, the local interpreter does not need to use them at all; hence there is no causal relationship between these inputs and the prediction.

The separation between query mechanism and value computation resembles the two-stream attention mechanism in XLNet (Yang et al., 2019), where a separate query stream is introduced in addition to the normal content stream to enable the usage of target position information while avoiding "cheating" to work with arbitrary generation factorization order. Recently, Russin et al. (2019) used word embeddings as content vectors, whereas the attention is computed based on the outputs of an LSTM network. This approach shares a similar goal with ours to separate syntax and semantics, but the transformer presents its unique challenges due to the existence of multiple layers and multi-headed attentions and its lack of recurrence.

3 | Background

Attention is a key tool in machine learning for NLP. Bahdanau et al. (2014) first used attention mechanism in an encoder-decoder model for machine translation. During decoding, the model uses attention to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word. The transformer uses attention mechanisms to draw global dependencies between input and output without making use of recurrence. In this work, we replace the standard soft attention with discrete attention by modeling the attention as a discrete categorical random variable. However, training latent variable models exactly is computationally intractable, and Monte-Carlo estimation using techniques such as REINFORCE is slow and fraught with variance issues as we cannot backpropagate through the samples. The Gumbel-Softmax estimator is an efficient gradient estimator that replaces the non-differentiable sample from a categorical distribution with a differentiable sample from the Gumbel-Softmax distribution (Jang et al., 2016; Maddison et al., 2016). In the following, I give background on attention, the transformer, and the Gumbel-Softmax approximator.

3.1 Attention

Attention, as its name suggests, is motivated by human attention to some extent. When we look at a scene, we expend different levels of attention to different regions of the scene. We view a particular region in high resolution and everything else in low resolution. We can shift our focus to different regions and perform the necessary inference.

Attention mechanisms can also be used to pick up relationships between words in sentences. For instance, in the sentence "Jambay gave his sister a birthday gift.", the word "his" refers to "Jambay," and having seen "gave" and "birthday" we expect a word such as "gift" to follow. Therefore, a contextual representation for "his" would most probably include information about "Jambay." If we were generating the sentence word by word, it would make sense for "gift" to be generated after attending to the words "gave" and "birthday."

Attention is a mechanism where the model makes predictions or infers an element such as a token in a sentence by selectively attending to different parts of the input. We take the sum of the input values weighted by a vector of attention weights and use this weighted average for the prediction. Attention weights are learned during training.

It is worth noting that although attention can be broadly interpreted as a vector of importance weights, it is not clear whether the magnitude of attention weights inherently reflects the relative importance of the corresponding inputs (Jain and Wallace, 2019).

3.1.1 Mathematical Formulation

In models based on attention, we generate and use three sequences,

$$\boldsymbol{Q} = \Big(\boldsymbol{q}_0, \boldsymbol{q}_1, ..., \boldsymbol{q}_m\Big)^\top, \boldsymbol{q}_i \in \mathbb{R}^{d_q}$$

$$\boldsymbol{K} = \Big(\boldsymbol{k}_0, \boldsymbol{k}_1, ..., \boldsymbol{k}_n\Big)^\top, \boldsymbol{k}_i \in \mathbb{R}^{d_k}$$

$$\boldsymbol{V} = \Big(\boldsymbol{v}_0, \boldsymbol{v}_1, ..., \boldsymbol{v}_n\Big)^\top, \boldsymbol{v}_i \in \mathbb{R}^{d_v}$$

We call these sequences query, key, and value, respectively.

$\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ can either be generated from a single input sentence $\boldsymbol{x}$ (self-attention) or $\boldsymbol{Q}$ can be generated from a target sequence $\boldsymbol{y}$ and $\boldsymbol{K}, \boldsymbol{V}$ from $\boldsymbol{x}$ (inter-attention). We explain the differences between the two methods in the next sub-section.

In attention, we use the queries $\boldsymbol{Q}$ and keys $\boldsymbol{K}$ to generate attention weights α and use these weights to combine the values $\boldsymbol{V}$ to generate a new sequence, usually called the context, $\boldsymbol{C} = \Big(\boldsymbol{c}_0, \boldsymbol{c}_1, ..., \boldsymbol{c}_m\Big)^\top, \boldsymbol{c}_i \in \mathbb{R}^{d_v}$. $\boldsymbol{C}$ is then used for further downstream tasks such as prediction. The following equations describe how $\boldsymbol{C}$ is generated:

$$\boldsymbol{c}_t = \sum_{i=0}^{n-1} \alpha_{t,i} \boldsymbol{v}_i$$

$$\alpha_{t,i} = \frac{\exp(\texttt{score}(\boldsymbol{q}_t, \boldsymbol{k}_i))}{\sum_{i'=0}^{n-1} \exp(\texttt{score}(\boldsymbol{q}_t, \boldsymbol{k}'_i))}$$

$\texttt{score}(\boldsymbol{q}_t, \boldsymbol{k}_i)$ is a measure of the match between query $\boldsymbol{q}_t$ and key $\boldsymbol{k}_i$. There are many different forms of the score function such as

- General: $\texttt{score}(\boldsymbol{q}_t, \boldsymbol{k}_i) = \boldsymbol{q}_t^\top \boldsymbol{W} \boldsymbol{k}_i$ where $\boldsymbol{W}$ is learned
- Dot-Product: $\texttt{score}(\boldsymbol{q}_t, \boldsymbol{k}_i) = \boldsymbol{q}_t^\top \boldsymbol{k}_i$
- Scaled Dot-Product: $\texttt{score}(\boldsymbol{q}_t, \boldsymbol{k}_i) = \frac{\boldsymbol{q}_t^\top \boldsymbol{k}_i}{\sqrt{n}}$

3.1.2 Inter-attention

Inter-attention is normally used to align two sequences, usually known as the source sequence $\boldsymbol{x}$ and the target sequence $\boldsymbol{y}$. For instance, in encoder-decoder[1] sequence-to-sequence[2] models used for machine translation (Bahdanau et al., 2014; Luong et al., 2015), the key and value vectors $\boldsymbol{k}, \boldsymbol{q}$ are generated from $\boldsymbol{x}$ after passing it through an encoder network. The query $\boldsymbol{q}_t$ at time t is generated by passing the target sequence up to and including index $t-1$, i.e. $\boldsymbol{y}_{0:t-1}$ through a decoder network. The context vector $\boldsymbol{c}_t$ obtained after attention is then used for predicting the next word (at time t) in the translation.

3.1.3 Self-attention

Self-attention is used to make a prediction for one part of a sequence $\boldsymbol{x}$ using other parts of the same sequence. In this case, all of $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$ are generated using $\boldsymbol{x}$. This type of attention can be used to encode a sequence to draw global information such as context or to generate sequences by attending to the words generated so far.

3.2 Transformer

The transformer is a transduction model[3] that uses self-attention to compute representations of its input and output without using recurrent neural networks or convolution. It has an encoder-decoder structure where it uses stacked self-attention and position-wise[4], fully connected layers for both the encoder and decoder. We note here that although the original transformer is a sequence to sequence model used for machine translation, its components have been used independently for different tasks. For instance, the encoder is used for language understanding and text classification (Devlin et al., 2018) while the decoder is used for decoder language modeling (Radford et al., 2019).

We describe the transformer model in the following sub-sections.[5]

[1] In an encoder-decoder model, an encoder network encodes an input $\boldsymbol{x}$ into some hidden representation $\boldsymbol{h}$. A decoder network uses $\boldsymbol{h}$ to generate output $\boldsymbol{y}$

[2] In a sequence-to-sequence model, the input $\boldsymbol{x}$ and the output $\boldsymbol{y}$ are both sequences.

[3] A transduction model transforms an input sequence into another form.

[4] Position-wise means it applies the same linear transformation to each element in the sequence

[5] I adapted Lilian Weng's post on the transformer at `https://lilianweng.github.io/lil-log/2020/04/07/the-transformer-family.html` along with the original transformer paper by Vaswani et al. (2017) for this overview.

3.2.1 Attention

The transformer uses scaled dot-product attention. The input consists of queries and keys of dimension d_k, and values of dimension d_v. For a query $\boldsymbol{q}_i$ and key $\boldsymbol{k}_j$, the attention score is given by

$$\alpha_{i,j} = \frac{\exp(\boldsymbol{q}_i^\top \boldsymbol{k}_j / \sqrt{d_k})}{\sum_{r \in S_i} \exp(\boldsymbol{q}_i^\top \boldsymbol{k}_r / \sqrt{d_k})}$$

where S_i is a collection of key positions for the i-th query to attend to.

In practice, the attention function is computed on set of queries simultaneously. We pack the queries into a matrix $\boldsymbol{Q}$ where the rows are the query vectors transposed. We similarly create a key matrix $\boldsymbol{K}$ and a value matrix $\boldsymbol{V}$. The attention matrix is then computed as

$$\texttt{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \texttt{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}})\boldsymbol{V}$$

where the softmax is over the elements of each row.

Multi-Head Self Attention Rather than only computing the attention once, the multi-head mechanism linearly projects the queries, keys, and values h times with different learned linear projections into d_k, d_k, and d_v dimensions respectively. It then computes the scaled dot-product attention over each of the h subspaces in parallel, leading to h sets of d_v-dimension output values. These outputs are concatenated and linearly transformed into expected dimensions.

$$\texttt{MultiHeadAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \texttt{concat}(\texttt{head}_1, \ldots, \texttt{head}_h)\boldsymbol{W}^O$$

where $\texttt{head}_i = \texttt{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V)$

$\boldsymbol{W}_i^Q \in \mathbb{R}^{d_{\texttt{model}} \times d_k}, \boldsymbol{W}_i^K \in \mathbb{R}^{d_{\texttt{model}} \times d_k}, \boldsymbol{W}_i^V \in \mathbb{R}^{d_{\texttt{model}} \times d_v}$, and $\boldsymbol{W}^O \in \mathbb{R}^{d_v \times d_{\texttt{model}}}$ are learned matrices.

3.2.2 Encoder-Decoder Architecture

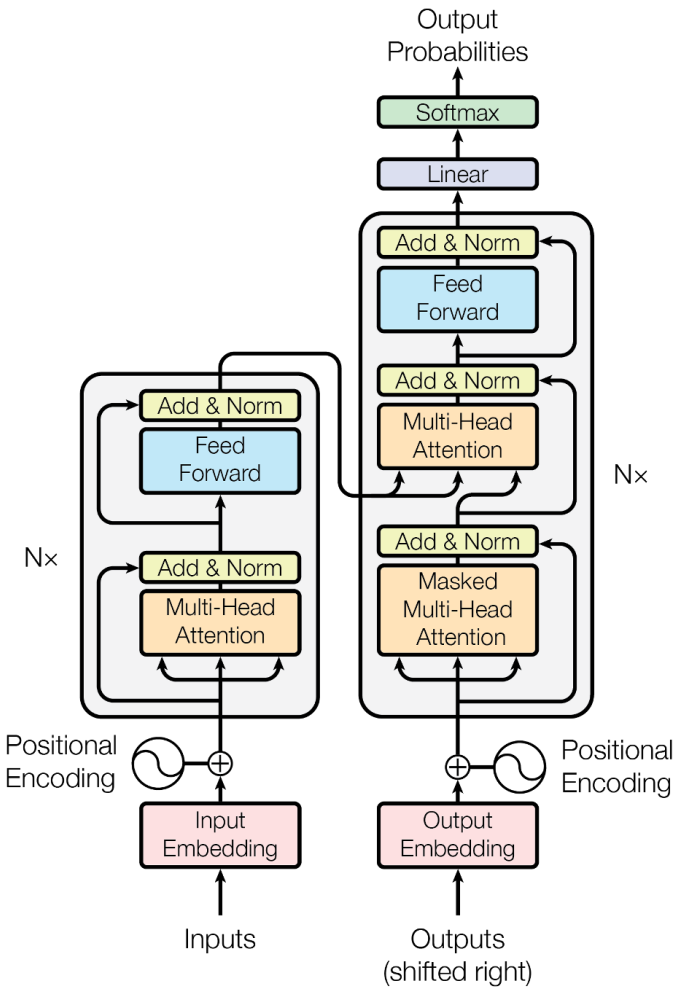


Figure 3.1: Transformer model architecture. Source: Figure 1 in Vaswani et al. (2017)

Encoder The encoder maps an input sequence of tokens $\boldsymbol{x} = (x_1, \ldots, x_T)$ into a sequence of vector representations $\boldsymbol{h} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T)$ using a stack of 6 identical layers as shown in Figure 3.1. The output a layer $l-1$ becomes the input of the layer l. Each of these layers is made up of two sub-layers - a multi-head self-attention layer followed by a position-wise fully connected feed-forward network. Each sub-layer has a residual connection[6] and layer normalization[7].

[6]Residual Connections (He et al., 2016) are of the form $\boldsymbol{y} = F(\boldsymbol{x}) + \boldsymbol{x}$ where $F(\boldsymbol{x})$ is a non-linear transformation of $\boldsymbol{x}$.

[7]Layer normalization (Ba et al., 2016) normalizes the inputs across the features.

Decoder The decoder of the transformer uses information from the encoder representation to generate the final output (for instance, translation). Its architecture is similar to the encoder, but it uses two multi-head attention sub-layers instead of one. The extra attention sub-layer performs multi-head attention over the output of the encoder stack. The self-attention sub-layer in the decoder stack is modified to prevent positions from attending to the future.

Simplified Notation For the sake of notational simplicity, we will use the following to describe the architecture. We elide layernorm, dropout, most residual connections, and multi-headedness, but they are all included in the final models.

We begin by encoding tokens x_i with a position-specific embedding function e [8] to vectors $\boldsymbol{h}_i^0$. Each layer of the transformer then produces new vectors under the following recursion:

$$\begin{aligned} \boldsymbol{B}_i &\leftarrow \text{FFN}(\boldsymbol{h}_i^{(l-1)}) \ \ \forall \ \ i \in 1, \ldots, n \\ \boldsymbol{K}, \boldsymbol{V}, \boldsymbol{Q} &\leftarrow (\boldsymbol{B}\boldsymbol{W}^{(K)}, \boldsymbol{B}\boldsymbol{W}^{(V)}, \boldsymbol{B}\boldsymbol{W}^{(Q)}) \\ \boldsymbol{A} &\leftarrow \texttt{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}) \\ \boldsymbol{h}_i^{(l)} &\leftarrow \boldsymbol{h}_i^{(l-1)} + \boldsymbol{A}_i \boldsymbol{V} \\ & \quad \forall \ \ i \in 1, \ldots, n \end{aligned} \tag{3.1}$$

Here l is the layer index, FFN is a feed-forward neural network, and $\boldsymbol{W}$ are learned projection parameters. At the final layer L we can make a prediction utilizing $\boldsymbol{h}^{(L)}$, e.g. a softmax over possible y labels to compute $p(y|\boldsymbol{x})$.

[8] Self attention is permutation invariant. The positional encoding provides order information to the model. This is done by adding a position-specific vector such as a sinusoidal encoding to the word embeddings of each token x_i

3.3 Gumbel Softmax Estimator

The Gumbel-Softmax distribution (Jang et al., 2016; Maddison et al., 2016) is a continuous distribution over the simplex that provides a continuous approximation to sampling from the categorical distribution. Given a categorical distribution (with K categories) defined by log probabilities $\boldsymbol{l}$, the Gumbel-Softmax generative process is defined by first sampling $U_k \sim \text{Uniform}(0, 1)$, and then returning a K-dimensional sample vector $\boldsymbol{s} \in \Delta^{K-1}$ where

$$s_k = \frac{\exp((l_k + g_k)/\tau)}{\sum_{j=1}^{K} \exp((l_j + g_j)/\tau)} \quad \text{for } k = 1, \dots K$$

where $g_k = -\log(-\log(U_k))$ is Gumbel noise and τ is a temperature parameter controlling the entropy of the distribution. As $\tau \to 0$, samples given by the Gumbel-Softmax distribution conforms to the same distribution as one-hot categorical samples.

The Gumbel-Softmax distribution is smooth for $\tau > 0$, and so the sample $\boldsymbol{s}$ is differentiable with respect to $\boldsymbol{l}$. We can thus use backpropagation to compute gradients during training. This procedure is called Gumbel-Softmax Estimator.

At non-zero temperature, the Gumbel-Softmax samples are, however, not identical to samples from the corresponding categorical distribution. The samples are close to one-hot at low temperatures, but the variance of the gradients is large. Meanwhile, at higher temperatures, the samples are smooth, but the variance of the gradients is small. To manage this tradeoff during training, we usually start at a high temperature and anneal to a low temperature.

4 | Proposed Model Architecture

The main processing work of the transformer happens in the feed-forward neural network layers which contain the majority of the non-embedding parameters. However, for these layers to be effective it is crucial that information from other tokens be aggregated together. The attention layer serves as the single source of this routing in the model.

Because attention is central for routing and determines the receptive field[1] of the transformer, it has been a main source of study for the transformer and related models. If attention can be understood, then, in theory, the interconnections between words can be mapped and perhaps even manipulated.

Unfortunately, much of this work has so far produced negative results (Kovaleva et al., 2019; Jawahar et al.). The underlying problem is that while any one attention layer may target only a small amount of keys, in aggregate repeated applications of multi-headed attention quickly connect every position to every other. Learned attention acts in a soft way and roughly pools together all elements into a vector. While this may be usable for high-level analysis, it does not allow us to truly separate out anything about the model decisions.

4.1 Discrete Attention Transformer

Several recent works (Deng et al., 2018; Shankar et al., 2018) have explored alternatives to soft attention for single-layer attention models. The goal has been to build models that can replace soft-attention with latent control variables that select a single position to use.

We apply this approach directly to the transformer. Formally, we modify the above deterministic Eq 3.1 with a single sampling step:

$$\begin{aligned} \boldsymbol{h}_i^{(l)} \quad \leftarrow \quad & \boldsymbol{h}_i^{(l-1)} + \boldsymbol{V}_{z_i^l} \text{ where } z_i^l \sim \texttt{Categorical}(\boldsymbol{A}_i) \\ & \forall \;\; i \in 1, \ldots, T \end{aligned}$$

[1] By receptive field, we mean the subset of the input sequence that affects the output

This change requires marginalization to compute a prediction, i.e. $p(\boldsymbol{y}|\boldsymbol{x}) = \sum_z p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x})$ since we do not observe $\boldsymbol{z}$. In a single layer model, this might be a tractable sum, but for transformer it is combinatorial. Even with single-headed attention it would require $O(T^L)$ time.

To avoid this issue at training, we use the Gumbel-Softmax estimator. However, when testing, we replace Gumbel-Softmax with the greedy argmax at each step.

Training in this way requires only minor modifications, but leads to different attention structure. Figure 4.1 shows an experiment demonstrating the model's ability to learn sharper dependencies on a simple short term language modeling task.

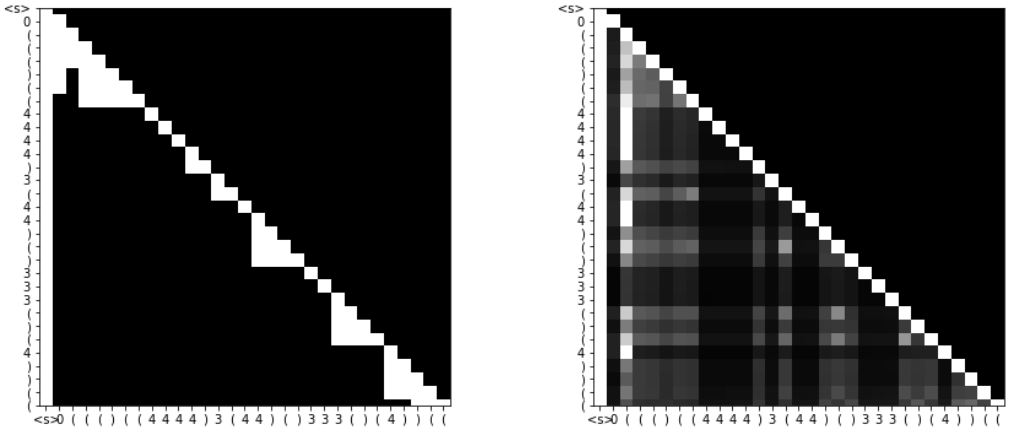


Figure 4.1: Synthetic variable distance language modeling experiment showing aggregate attention of (left) discrete transformer (right) transformer under a sparsity regularizer. (Experimental details in Appendix.)

4.2 Controller / Model Separation

Due to the hard decision in discrete attention, model predictions are made based on a well-defined pathway from the original words. This is a key property for model interpretability; however, it also means that internal routing decisions go through this hard bottleneck when it might be better for the router to have more contextual information (through soft attention).

We consider an extension that avoids this issue while maintaining the key benefit of discrete transformer. We propose separating out the routing controller from the core model. This controller can use soft attention for its internal computation, but only interacts with the full model to make hard routing decisions. Only the

model stream representations are used for downstream task.

The architecture for this extension is shown in Figures 4.2. Let the controller representation at timestep i in layer l be $\boldsymbol{g}_i^{(l)}$ and the model representation be $\boldsymbol{h}_i^{(l)}$. The first layer representations are set to the corresponding word embeddings. Intuitively, in this model, values are computed only by the model network, keys and queries are computed only by the controller network, and attention is shared.

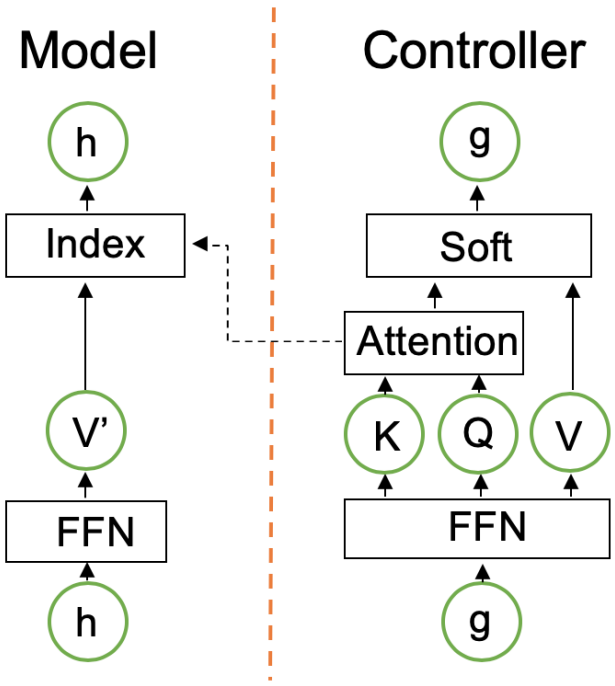


Figure 4.2: Two-Stream Discrete Transformer architecture. Controller stream computes the attention distribution which is used to produce next hidden states while also constructing the model architecture through latent hard attention.

For each attention layer l, the two-streams are updated as follows:

Model Stream	**Controller Stream**
$\boldsymbol{C}_i \leftarrow \text{FFN}(\boldsymbol{h}_i^{(l-1)})$	$\boldsymbol{B}_i \leftarrow \text{FFN}(\boldsymbol{g}_i^{(l-1)}) \;\; \forall \;\; i \in 1, \ldots, T$
$\boldsymbol{V}' \leftarrow \boldsymbol{C}\boldsymbol{W}^{(h)}$	$\boldsymbol{K}, \boldsymbol{V}, \boldsymbol{Q} \leftarrow (\boldsymbol{B}\boldsymbol{W}^{(K)}, \boldsymbol{B}\boldsymbol{W}^{(V)}, \boldsymbol{B}\boldsymbol{W}^{(Q)})$
	$\boldsymbol{A} \leftarrow \texttt{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}})$
$\boldsymbol{h}_i^{(l)} \leftarrow \boldsymbol{h}_i^{(l-1)} + \boldsymbol{V}'_{z_i^l}$ where $z_i^l \sim \text{Cat}(\boldsymbol{A}_i)$	$\boldsymbol{g}_i^{(l)} \leftarrow \boldsymbol{g}_i^{(l-1)} + \boldsymbol{A}_i\boldsymbol{V} \;\; \forall \;\; i \in 1, \ldots, T$

Note that the controller stream is completely independent of the model stream and could even be computed before hand. Furthermore, assuming a fixed controller distribution, the model stream becomes a vanilla feedforward network. In practice, the two streams distributions can be trained together as a single network. We evaluate the interpretability and performance of the proposed model in the following chapters.

Additional benefits of the separation The two-stream architecture was motivated by Russin et al. (2019) that uses a distinct parser and a composition function to solve a synthetic dataset called ListOps. The parser generates a tree structure for a sequence while the composition function follows the tree generated by the parser to produce a sequence representation. The two components are trained together.

The controller stream in our architecture corresponds to the parser, while the model stream corresponds to the composition function. This analogy suggests that in addition to allowing the use of soft attention in the controller stream, the two-stream architecture could enable the separation of model concerns between the streams, in particular utilizing highly-syntactic representation for the controller stream. The controller stream only gets training signals through the attention weights $\boldsymbol{A}$, i.e., from how well it combines the model representations.

The two sentences "Jambay walked to his home" and "Kinley ran to her school" have the same syntactic structure. In a task such as translation, given a correct attention distribution (same for both sentences), a well-trained model stream should produce correct outputs that differ only in meaning and not structure (here we assume that their translations also have the same structure as each other). It is sufficient for the controller stream to know that the sentences are of the form "Noun Verb Preposition Possessive_Pronoun Noun".

It is thus possible that the controller stream picks up syntactic structures as it learns how to compose the model representations. We indeed observe this phenomenon in our experiments.

5 | Experimental Setup

5.1 Tasks

Text Classification Given a sentence $\boldsymbol{x}$, the goal is to predict a class y. We use this task to evaluate the performance and interpretability of the model.

Machine Translation Given a sentence $\boldsymbol{x}$ in a source language, the goal is to its translation in the target language. We use this task to evaluate the performance and interpretability of the model.

Multiview Arithmetic Evaluation This is formulated as a classification task where given an arithmetic expression $\boldsymbol{x}$, the goal is to predict the value y of the expression. We have two parallel corpora for prefix and infix notations of arithmetic expressions. We call this multiview since the two different notations can be thought of as viewing the same underlying expressions in different ways. This task is used to explore the application of the two-stream architecture in multiview learning, where we simultaneously train on both views for improved performance.

5.2 Data

We use several benchmark datasets for text classification and machine translation. We generate a synthetic dataset for multiview arithmetic evaluation.

Text Classification Following Jain and Wallace (2019), we use the following binary classification datasets:

Stanford Sentiment Treebank (SST) (Socher et al., 2013). This dataset consists of 10,662 sentences tagged with sentiment on a scale from 1 (most negative) to 5 (most positive). Neutral (3) sentences are filtered out and the remaining sentences are made re-classified as positive (4,5) or negative (1,2).

IMDB Large Movie Reviews Corpus (IMDB) (Maas et al., 2011). This dataset consists of 50,000 polarized (positive or negative) movie reviews. It is split into half for training and testing.

20 Newsgroups (Hockey vs Baseball)[1]. The 20 Newsgroups dataset consists of 20,000 newsgroup correspondences, divided into 20 categories with a nearly even distribution. We use the instances belonging to the baseball and hockey categories.

AG News Corpus (Business vs World)[2]. The AG News Corpus consists of 496,835 news articles from 2000+ sources. We use the articles belonging to world and business categories.

Machine Translation We conduct experiments on two machine translation datasets:

IWSLT (Cettolo et al., 2014). This is a standard small-scale English-German benchmark consisting of around 160000 training examples (pairs of English and German sentences) and 7000 test examples. The shared source-target vocabulary is made of about 10000 tokens.

WMT 2014 (Bojar et al., 2017). This is a larger English-German benchmark consisting of about 4.5 million sentence pairs. Sentences were encoded using byte-pair encoding. The shared source-target vocabulary is made of about 35700 tokens.

Multiview Arithmetic Evaluation We adapt the arithmetic reasoning dataset from Kim et al. (2017) to a multi-view classification setting, where the same math expression can be expressed in both prefix order and infix order.

For instance, the below is a math expression in infix order:

$$((4 + 1) * 3) + (2 + 4)$$

The same expression can be represented in the prefix order:

$$(+ (* (+ 4 1) 3) (+ 2 4))$$

We limit the operations to multiplication * and addition +. We also limit the depth of nesting to 3. The example above has depth 2 nesting.

To keep the number of classes bounded, we restrict the numbers to the set $\{0, 1, 2, 3, 4\}$, and evaluate the expressions modulo 5 such that the above example evaluates to 1. This creates a 5-class classification problem. We can perform the modulo operation without breaking the structure of the problem since (A + B) mod C = A mod C + B mod C and (A * B) mod C = A mod C * B mod C.

We generate 30000 training samples and 3000 test samples with the samples evenly distributed among the depth levels.

[1] http://qwone.com/~jason/20Newsgroups/

[2] http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

5.3 Model Architectures and Settings

We use standard transformer architectures in all experiments. We implement our models based on Fairseq (Ott et al., 2019).

Text Classification We use the transformer encoder. Each sentence $\boldsymbol{x}$ is prepended with a special token <cls> and fed through the encoder to obtain $\boldsymbol{h}^{(L)}$. We use a learned linear layer to project $\boldsymbol{h}_0^{(L)}$ to a 5-dimensional space and pass it through softmax to get a distribution $\tilde{\boldsymbol{y}}$ over the classes. During training, we optimize the Cross-Entropy loss. At test time, we predict the class by taking argmax over the final outputs.

We consider two models: Transformer, standard soft-attention transformer model; and Discrete Transformer: two-stream controller / model separation.

We use $d_{\text{model}} = 64$, $d_{\text{ff}} = 128$, $L = 6$, $h = 4$ (we refer to Vaswani et al. (2017) for hyperparameter definitions).

The Gumbel temperature is set to 1 throughout this paper.

Machine Translation We use the standard encoder-decoder architecture. Our Discrete Transformer uses encoder and decoder stacks analogous to those in Transformer.

For encoder-decoder attention, the controller stream of the encoder provides the keys $\boldsymbol{K}$ and the values $\boldsymbol{V}$, the encoder's value stream provides the values $\boldsymbol{V}'$, and the decoder's controller stream provides the queries $\boldsymbol{Q}$.

We consider three models: Transformer, standard soft-attention transformer model; Single-Stream Discrete Transformer, transformer with discrete attention; and Discrete Transformer: two-stream controller / model separation.

For WMT, we use the base model with $d_{\text{model}} = 512$, $d_{\text{ff}} = 2048$, $L = 6$, $h = 8$. For IWSLT, we use $d_{\text{model}} = 512$, $d_{\text{ff}} = 1024$, $L = 6$, $h = 4$ since it is more prone to overfitting.

Multiview Arithmetic Evaluation Similar to text classification, we use the transformer encoder followed by a linear projection layer.

We consider two models: Two-Stream Transformer, two-stream controller / model separation with soft attention[3]; and Multiview Two-Stream

[3]We use soft attention due to issues with training a discrete attention model with good accuracy. This experiment is used to demonstrate the advantage of the two-stream architecture so using soft-attention doesn't affect the results.

TRANSFORMER, modified TWO-STREAM TRANSFORMER with two controller streams and a model stream.

A separate TWO-STREAM TRANSFORMER is trained for each of the infix and prefix views.

The MULTIVIEW TWO-STREAM TRANSFORMER is trained on the infix and prefix views simultaneously. For a give pair of expressions $(\boldsymbol{x}_{\text{infix}}, \boldsymbol{x}_{\text{prefix}})$, we use a separate controller stream for each of $\boldsymbol{x}_{\text{infix}}$ and $\boldsymbol{x}_{\text{prefix}}$ and share the same model stream. We make two corresponding predictions $\tilde{\boldsymbol{y}}_{\text{infix}}$ and $\tilde{\boldsymbol{y}}_{\text{prefix}}$ and optimize for the Cross-Entropy Loss for each. The motivation here is that given the correct expression tree, evaluation is the same for both views. The numbers and the operations have the same meaning in both. Thus if the controller streams learn the correct attention decisions, the model stream should perform the same computations for both views.

We use $d_{\text{model}} = 64$, $d_{\text{ff}} = 128$, $L = 6$, $h = 4$.

5.4 Metrics

Model Performance We evaluate the performances of the models by computing accuracy for the text classification tasks and BLEU score[4] for machine translation.

Model Interpretability To quantify model interpretability, we follow Jain and Wallace (2019) and use the correlation between attention weights and gradient-based inputs relevance measure, i.e. whether attention corresponds to first-order impact of words.

We measure the importance of the i-th token x_i as $\|\frac{\partial \hat{y}}{\partial \mathbf{h}_i^0}\|_2$ where $\hat{y}$ is the model's prediction.

Since transformers have multiple attention layers, we add up all attentions to get a sequence $a_1, \cdots, a_T$. This serves as an approximation for the amount of attention paid by the model to each of the input tokens when computing $\hat{y}$.

For every example in the dataset, we compute the Kendall τ rank correlation between a_i and $\|\frac{\partial \hat{y}}{\partial \mathbf{h}_i}\|_2^0$. The Kendall τ ranking correlation measures the agreement over two rankings and has a value of 1 when the agreement is perfect.

[4]BLEU Score (Papineni et al., 2002) is an automatic machine translation evaluation that correlates highly with human evaluation. It is the standard metric used to evaluate machine translation.

6 | Results and Analysis

6.1 Text Classification

Figure 6.1 compares the histogram of the correlations of the Transformer and the Discrete Transformer on 20 News, and Table 6.1 presents the mean/standard deviation on multiple datasets.

We find that Discrete Transformer exhibits on average higher correlation than its soft counterpart Transformer, at the cost of slightly worse performance on some benchmarks.

Jain and Wallace (2019) showed that in soft attention models, the attention weights (nonzero everywhere) are not very correlated with the actual importance for the prediction. We observe the same phenomenon in Transformer.

However, Discrete Transformer correlates better - the model attends more to the relevant features/tokens, which is not very unexpected since in Discrete Transformer, tokens never attended to are not used at all and would not affect the final prediction. This is unlike soft attention where every token is assigned a nonzero weight, and the model can rely on a token with a small attention weight heavily.

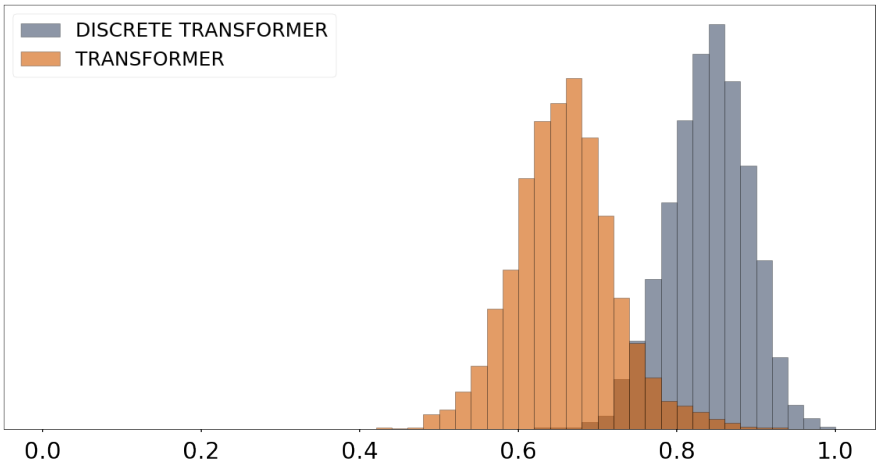


Figure 6.1: Histogram of attention correlations for text classification on AG News.

Dataset	Transformer	Disc Transf
SST	0.69±0.11 (79.6)	0.71±0.09 (76.1)
IMDB	0.64±0.05 (87.8)	0.79±0.03 (83.2)
20 News	0.67±0.11 (88.2)	0.72±0.07 (89.3)
AG News	0.66±0.06 (96.3)	0.84±0.05 (94.3)

Table 6.1: Text Classification Results. Mean and standard deviation of correlations on text classification datasets. The original task accuracy (%) is shown in parentheses.

6.2 Machine Translation

Table 6.2 shows the BLEU scores of the models along with attention correlations. Our baseline soft transformer performs identically to the published results. Single-Stream Discrete Transformer performs slightly worse than the soft counterpart Transformer by 0.7 BLEU points on WMT. On the other hand, our Discrete Transformer achieves similar performance as the soft baseline, showing that the soft controller stream helps alleviate model expressivity issues. The results are reversed on the much smaller IWSLT dataset, with the single-stream model performing better and nearing the results of the soft model, possibly due to overfitting in the two-stream model.

The correlation results further confirm that Discrete Transformer achieves better correlations than Transformer at a similar performance. Single-Stream Discrete Transformer also show higher correlations, but at a slightly worse performance.

Model	WMT	IWSLT
Vaswani2017 (base)	27.3	-
Transformer	27.2 (0.68±0.07)	28.7
S S Discrete Transformer	26.1 (0.75±0.06)	28.5
Discrete Transformer	26.5(0.78±0.05)	28.0

Table 6.2: WMT English-German (En-De) and IWSLT14 Results. For WMT mean and standard deviation of correlations are shown in parentheses.

Stream Analysis We observe that the model learns to separate out syntactic from semantic properties in its two streams.

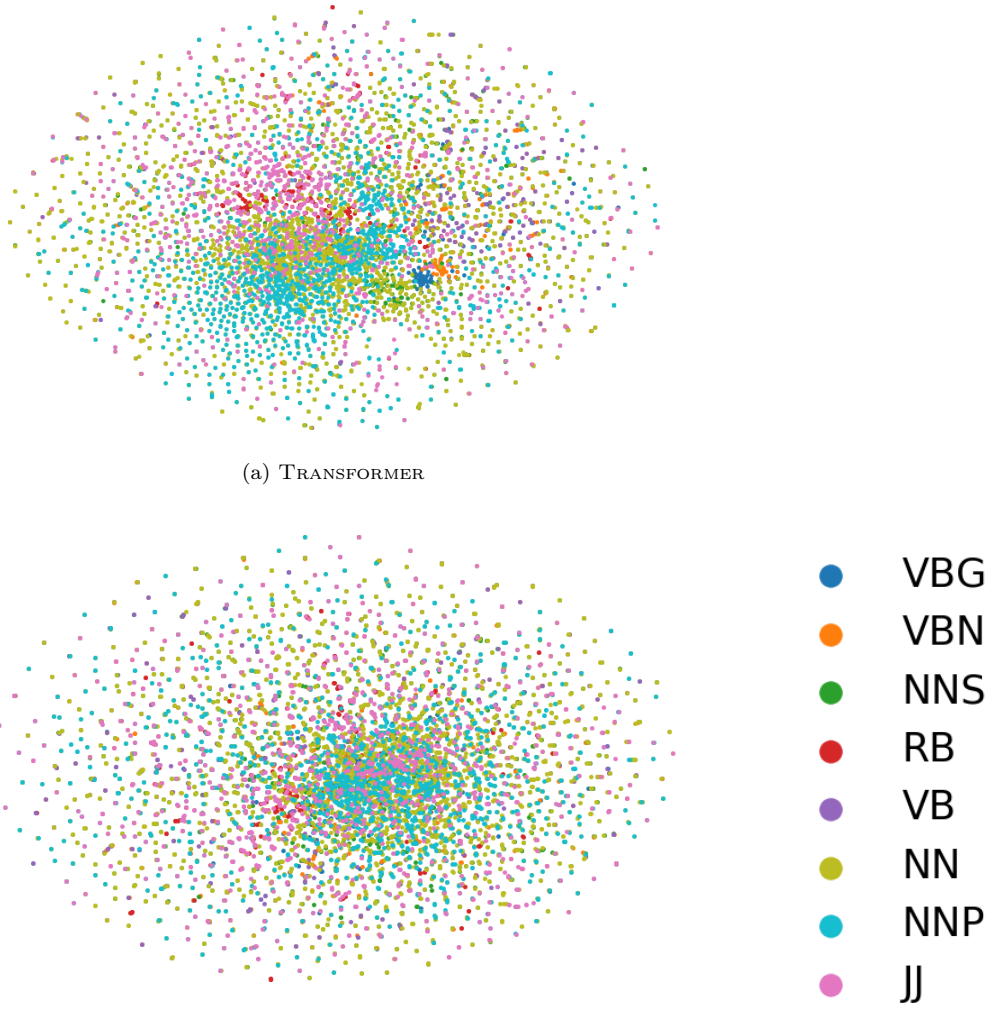
Model	Nearest Neighbors
TRANSFORMER	somewhat, slight, little, easily, differently, modest, bit, easy, briefly, rather
SS DISCRETE TRANSFORMER	somewhat, slight, briefly, little, bit, easily, minor, minimal, barely, partly
DISCRETE TRANSFORMER controller	somewhat, twice, little, substantially, constantly, almost, considerably, partly, easily, largely
DISCRETE TRANSFORMER model	somewhat, slight, minor, mild, light, little, easily, growth, significantly, Light

Table 6.3: Nearest neighbors of word "slightly" using internal representations from different modules. Colors mark POS tags (same legends as Figure 6.2)

For example, Figure 6.2 shows the t-SNE projections for word embeddings. Embeddings from the controller stream cluster by part-of-speech (POS) tags while those from the model stream do not. The qualitative nearest neighbors example in Table 6.3 further confirms this.

To quantify the difference in learned representations, we consider utilizing the learned representations. We experiment with using different source encoders from the WMT model as pre-trained representations for performing a syntactic chunking task from the CoNLL-2000 dataset. Syntactic chunking consists of dividing a text into syntactically correlated parts of words. For this task, we use the pre-trained encoder to obtain some vector representation of the source sentence, which is then passed through a linear layer to project it to the space of chunk types. The encoder is frozen, and only the linear projection layer is trained. For words broken into multiple tokens using the BPE tokenization, we use the vectors from the first token.

Table 6.4 shows the results from different models. The baseline result was obtained by selecting the chunk tag, which was most frequently associated with the current part-of-speech tag. Both discrete models outperform a TRANSFORMER, and but the controller stream of the DISCRETE TRANSFORMER does the best. Interestingly the model stream, which does not have to determine word relations, performs worse than even the baseline model.



(a) Transformer

(b) Discrete Transformer Model

(c) Discrete Transformer Controller

Figure 6.2: t-SNE embedding plot colored by POS tag on the WMT train corpus. Best viewed in color.

Model	accuracy	precision	recall	F_1
Baseline	-	72.58	82.14	77.07
Transformer	89.83	82.25	87.58	84.83
SS Disc Transformer	90.70	83.78	88.45	86.05
D Transformer controller	91.80	85.39	89.51	87.40
D Transformer model	83.17	73.07	81.28	76.96

Table 6.4: CoNLL-2000 Chunking. All models are trained with a linear projection over the final encoder layer of fixed WMT model.

Model	all depth	depth 1	depth 2	depth 3
TS Transformer prefix	45.9	0.75	33.6	29.2
TS Transformer infix	66.0	97.6	63.7	36.8
Multiview TS Transf prefix	65.3	97.9	62.6	35.5
Multiview TS Transf infix	68.3	97.9	66.9	40.1

Table 6.5: Accuracy on multiview arithmetic dataset broken down by depth

6.3 Multiview Arithmetic Evaluation

Table 6.5 shows the accuracy of the models for the two views. Multiview Two-Stream Transformer trained on both prefix and infix views gets a comparable performance on infix order, but it performs much better on prefix ordering compared to Two-Stream Transformer, showing its ability to transfer knowledge learned from infix view to prefix view. We speculate that this could be because the model stream, which is learned well through the infix view, guides the controller stream for prefix view. Moreover, the use of the same model stream makes it more robust as it needs to deal with two different syntactic structures.

This suggests that the two-stream architecture can be used for multiview learning by simultaneously learning multiple views through a shared stream.

7 | Conclusion

In this work, we present discrete transformer, a modification to the transformer with discrete attention, and separate routing. Through experiments on several benchmarks datasets for text classification and machine translations, we show that the model is able to maintain similar levels of performance as the standard transformer. Our analysis using correlations between model attention and gradient-based feature importance shows that the model makes more interpretable decisions.

Also, we discover that the two-stream architecture enables the model to separate out syntactic properties. We further explore the benefits of this separation by using the two-stream architecture in a type of multiview learning. We show that training on two views of arithmetic expressions simultaneously with a shared controller stream improves model performance.

7.1 Future Work

This style of model opens up the potential for many possible experiments in NLP.

Because the model makes hard intermediary decisions, the semantic model can be shown to only depend on a subset of the data. This property could be used to check for or remove bias from a model, for instance, to ensure that the production of gendered pronoun does not depend on spurious context.

Similarly, because the dependencies (attentions) are predicted separately, additional priors or regularization could be used to enforce specific syntactic structures. We describe such an experiment in the Appendix. Using a sparsity regularizer, our model recovers the true dependencies in a synthetic language modeling dataset without direct supervision (using the receptive fields): we get precision of 0.959 and recall of 0.92. However, in soft attention models, since attention weights in every layer are nonzero everywhere, every final prediction depends on all previous words.

There has been significant recent work on the separation of syntax and semantics (Chen et al., 2019; John et al., 2018; Russin et al., 2019). Our architecture and

results provide a potential avenue for similar work in transformer-based models.

Finally, there is room for the application of multiview learning on real-world tasks and datasets.

References

David Alvarez-Melis and Tommi S Jaakkola. 2017. A causal framework for explaining the predictions of black-box sequence-to-sequence models. *arXiv preprint arXiv:1707.01943*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450,*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ondřej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, and Julia Kreutzer. 2017. Proceedings of the second conference on machine translation. In *Proceedings of the Second Conference on Machine Translation*. Association for Computational Linguistics.

Toon Calders, Asim Karim, Faisal Kamiran, Wasif Ali, and Xiangliang Zhang. 2013. Controlling attribute effect in linear regression. In *2013 IEEE 13th International Conference on Data Mining*, pages 71–80. IEEE.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730. ACM.

Mauro Cettolo, Jan Niehues, Sebastian Stuker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *Proceedings of IWSLT*.

William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. 2015. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. A multi-task approach for disentangling syntax and semantics in sentence representations. In *NAACL-HLT*.

Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. 2016. Retain: An interpretable predictive model

for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*, pages 3504–3512.

Gonçalo M Correia, Vlad Niculae, and André FT Martins. 2019. Adaptively sparse transformers. *arXiv preprint arXiv:1909.00015*.

Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M Rush. 2017. Image-to-markup generation with coarse-to-fine attention. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 980–989. JMLR. org.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. 2018. Latent alignment and variational attention. In *Advances in Neural Information Processing Systems*, pages 9712–9724.

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*.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, page 770–778.

Sarthak Jain and Byron C. Wallace. 2019. Attention is not explanation. *CoRR*, abs/1902.10186.

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

Ganesh Jawahar, Benoît Sagot, Djamé Seddah, et al. What does bert learn about the structure of language?

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2018. Disentangled representation learning for non-parallel text style transfer. In *ACL*.

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M Rush. 2017. Structured attention networks. *arXiv preprint arXiv:1702.00887*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. *arXiv preprint arXiv:1908.08593*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*.

Gilles Louppe. 2014. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

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*.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL-2002: 40th Annual meeting of the Association for Computational Linguistics*, page 311–318.

Ben Peters, Vlad Niculae, and André FT Martins. 2018. Interpretable structure induction via sparse attention. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 365–367.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Alvin Rajkomar, Eyal Oren, Kai Chen, Andrew M Dai, Nissan Hajaj, Michaela Hardt, Peter J Liu, Xiaobing Liu, Jake Marcus, Mimi Sun, et al. 2018. Scalable and accurate deep learning with electronic health records. *NPJ Digital Medicine*, 1(1):18.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the*

22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 1135–1144. ACM.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Jake Russin, Jason Jo, and Randall C O'Reilly. 2019. Compositional generalization in a deep seq2seq model by separating syntax and semantics. *arXiv preprint arXiv:1904.09708*.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626.

Shiv Shankar, Siddhant Garg, and Sunita Sarawagi. 2018. Surprisingly easy hard-attention for sequence to sequence learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 640–645.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. 2017. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE transactions on visualization and computer graphics*, 24(1):667–676.

Inigo Jauregi Unanue, Ehsan Zare Borzeshi, and Massimo Piccardi. 2018. A shared attention mechanism for interpretation of neural automatic post-editing systems. *arXiv preprint arXiv:1807.00248*.

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 *Proceedings of NIPS*.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Appendix

Dependency Learning On A Synthetic Language modeling Task

To test whether the sparse attentions learned by our approach correspond to true underlying dependencies, we adapt a synthetic stack language dataset from Strobelt et al. (2017) where we know the true dependencies.

Data The vocabulary of the language consists of $\{0, 1, 2, 3, 4, (,)\}$, and the language must match parentheses. Numbers are emitted randomly, but must match the nesting level (the number of open left parentheses). Nesting is limited to depth 4. We follow this grammar and created a training set of 50k sequences, validation/test sets of 5k/5k sequences, with sequence length being 30. We train models to do language modeling on this dataset, where the true dependency for a given target word is the span between the last number and the previous word.

Sparsity Regularization Our goal on this dataset is to recover the true dependencies for any given word in an unsupervised way. To get a trade-off between precision and recall, we apply a sparsity regularizer. Here we consider a way to penalize the size of the receptive field at the final layer $|\bigcup_i r(i, L)|$. We want a differentiable version of $|\bigcup_i r(i, L)|$. Let's use s_{ij}^l to denote whether the representation of token i relies on embedding of token j at layer l, i.e. $j \in r(i, l)$, then we have the recursion that

$$s_{ij}^l = \min\left(s_{ij}^{l-1} + \sum_k z_{ik}^l s_{kj}^{l-1}, 1\right)$$

Where the internal representation of token i depends on embedding of token j if $s_{ij}^{l-1} = 1$ (due to residual connections, the dependencies of a layer below are also the current dependencies) or if i attends to k at layer $l-1$ and $s_{kj}^{l-1} = 1$ (the dependencies of the token attended to also become the current dependencies). Based on this recursion (and initial conditions that $s_{ij}^0 = 1(i = j)$), the final layer

receptive field size can be calculated as $|\bigcup_i r(i, L)| = \sum_i \min\left(\sum_j s_{ij}^L, 1\right)$. We note that since during training z comes from the Gumbel-Softmax, the z_{ik} values are computed as a soft approximation (as opposed to indices). Therefore it provides useful gradients and can be directly applied as a regularizer.

Experiments On this dataset, we train SINGLE STREAM DISCRETE TRANSFORMER with proper sparsity regularization (coefficient 0.1). At test time, we use argmax to get discrete attentions, and aggregate attentions to get the receptive field of each prediction. We were able to get precision of 0.959 and recall of 0.920 compared to the ground truth dependencies. In Figure 4.1, we show an example of the learned receptive field versus the ground truth dependency. On the other hand, if we aggregate attentions of a soft TRANSFORMER model via Eq. 7.1, the result is much messier, even with the attention sparsity regularizer.

We use $d_{\text{model}} = 64$, $d_{\text{ff}} = 256$, $L = 4$, $h = 2$.