Interpreting Attention-Based Models for Natural Language Processing

Steven J. Signorelli Jr
Steven.J.Signorelli.Jr.21@Dartmouth.edu

INTERPRETING ATTENTION-BASED MODELS FOR NATURAL LANGUAGE PROCESSING

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Bachelor of Arts

in

Computer Science

by

S. Joseph Signorelli Jr.

Advised by Professor Soroush Vosoughi

DARTMOUTH COLLEGE

Hanover, New Hampshire

June 1, 2021

Abstract

Large pre-trained language models (PLMs) such as BERT and XLNet have revolutionized the field of natural language processing (NLP). The interesting thing is that they are pre-trained through unsupervised tasks, so there is a natural curiosity as to what linguistic knowledge these models have learned from only unlabeled data. Fortunately, these models' architectures are based on self-attention mechanisms, which are naturally interpretable. As such, there is a growing body of work that uses attention to gain insight as to what linguistic knowledge is possessed by these models. Most attention-focused studies use BERT as their subject, and consequently the field is sometimes referred to as BERTology. However, despite surpassing BERT in a large number of NLP tasks, XLNet has yet to receive the same level of attention (pun intended). Additionally, there is an interest in their field in how these pre-trained models change when fine-tuned for supervised tasks. This paper details many different attention-based interpretability analyses and performs each on BERT, XLNet, and a version of XLNet fine-tuned for a Twitter hate-speech-spreader detection task. The purpose of doing so is **1.** to be a comprehensive summary of the current state of BERTology **2.** to be the first to do many of these in-depth analyse on XLNet and **3.** to study how PLMs' attention patterns change over fine-tuning. I find that most identified linguistic phenomenon present in the attention patterns of BERT are also present in those of XLNet to similar extents. Further, it is shown that much about the internal organization and function of PLMs, and how they change over fine-tuning, can be understood through attention.

Preface

As I reflect on the end of my undergraduate education and my time at Dartmouth in general, I don't think I could have picked a better capstone to a time that has brought me so much growth and personal satisfaction; I used every skill and quality I previously possessed, and then some, to complete this project, and I know it will forever be something I am proud of.

However, I would not have been able to complete this without the support of a number of amazing people for whom I am incredibly grateful. First, I would like to thank my advisor Professor Soroush Vosoughi whose machine learning course inspired me to progress on the academic track that has culminated in this project, and who took me on as an advisee when he was already at capacity, yet I was never able to tell that this was the case. I would also like to thank Weicheng Ma who helped to get me started in the field and to develop GROVER, and who was always quick to reply to questions about bugs I could not figure out.

I would like to thank my parents for everything, but with respect to this project specifically, I would like to thank them for inspiring in me a love of learning and a strong work ethic, and for showing me what can be achieved through the combination of the two.

Finally, I would like to thank Keurig, Advil, and Two Friends' Big Bootie Mixes for giving me the energy to work on this project even when I did not have it.

Contents

Chapter 1

Introduction

Section 1.1

Motivation

Machine learning is becoming increasingly embedded into our society. From healthcare to automation to influencing the content we are exposed to on a daily basis, advances in machine learning techniques and models have allowed machine learning to play larger roles in more and more aspects of our lives. However, as machines are given autonomy to make real-world decisions with increasing consequence, the need to understand the inner workings of these machines also increases. While this is important, it is often difficult to do; trained models are frequently viewed as a black-box, accepting an input on one end and spitting out an output on the other, with little insight as to how that output was generated from the input. In low-consequence situations such as a streaming service recommending content based on your viewing history, it is acceptable that the model remains a black-box as long as it is producing good results. But in situations where the decisions of a model are highly consequential, such as autonomous driving, parole decisions, and credit scoring, it is no longer sufficient to just know that the model works well; it then becomes important to learn what the model "knows." This is the primary concern of the field of machine learning interpretability: to know what a model knows. Interpretability is related to, but distinct from, explanation, which seeks to know what about *specific* inputs contribute to the corresponding output. Explanation is on an input-by-input basis, whereas interpretation is a more holistic view of the model and its behavior.

There are a two types of reasons for why it is important to know what a model knows. The first is the positive case: when a model becomes good at a task, we want to extract the knowledge that the model has learned that makes it perform so well. This is particularly important when a model begins to outperform humans at a task. For example, a deep learning model has achieved state-of-the-art breast cancer detection in mammograms, outperforming five out of five full-time breast-imaging specialists with an average increase in sensitivity of 14% (Lotter et al., 2021). While it is fantastic that this model is able to detect breast cancer earlier and more accurately than humans (and consequently lower mortality rate), it is still important to learn what this model knows that human specialists do not. Perhaps this novel knowledge can help inform future breast cancer research on what the early signs and causes of breast cancer may be. In general, interpretability allows machine learning to facilitate *human* learning. Christoph Molnar puts it well when he says that the "goal of science is to gain knowledge, but many problems are solved with big datasets and black box machine learning models. The model itself becomes the source of knowledge instead of the data. Interpretability makes it possible to extract this additional knowledge captured by the model" (Molner, 2019). Even more general than that, it is human nature to want to know the "why" - to break open the black-box and gain a better understanding of the world around us.

The second type of reason for why interpretability is important is the negative case: we want to confirm that a trained model has not learned anything bad or failed to learn something important. This case in particular becomes increasingly important as machine learning models are applied to greater tasks, especially those that require safety measures and testing. For example, cars go through extensive safety testing before they are allowed on the road. When a car uses a model for autonomous driving, that model then becomes part of the car's safety profile. How the model operates dictates how safe the car is for both the people in the car and those outside of it, and therefore it must be tested. But how does one do a thorough test of such a model? It is not enough to see that it performs well in a test environment because the model will be exposed to situations novel to it once out in the real world. One needs to be sure that the general knowledge possessed by the model includes everything it is supposed to (such as to stop at red lights) and nothing that it should not (such as to swerve into other traffic to avoid small animals in the road). Testing such as this requires understanding the knowledge possessed by the model, and as such requires interpretability. Additionally, machine learning models are only as good as the data on which they are trained. Consequently, any biases present in a model's training data will also be present in a model's actions. For example, it was found that a model used to predict how likely a parole candidate was to commit another crime (and thus inform their parole or prison sentence) was biased against black defendants even though race was not an explicit input to the model (Larson et al., 2016). Interpretability is needed to detect whether such biases are present in a model.

Section 1.2 Pre-Trained Language Models

As machine learning models grow in capability and application, the need for strong interpretability techniques will also grow. One driver of such growth has been a mechanism called attention, which was proposed by Bahdanau et al. in 2015. Models built with attention mechanisms have been successfully applied to a wide range of tasks such as speech recognition (Chorowski et al., 2015) and image captioning (Anderson et al., 2018). But the area that has seen some of the largest benefits from attention is that of natural language processing (NLP). Before attention, the dominant models for sequence transduction tasks such as NLP used recurrence or convolutional mechanisms. When attention was first introduced, it was added as a feature to these recurrent or convolutional neural networks to achieve even better results. However, attention has proven to be such a powerful mechanism that a new state-of-the-art was reached by doing away completely with recurrence and convolution and instead building a model based solely on attention. The attention-only model was first proposed in the paper "Attention Is All You Need" (Vaswani et al., 2017), and they named the attention-only architecture they built the Transformer, which achieved then-state-of-the-art performance on English-to-German and English-to-French translation tasks. Since then, Transformer-based models have been the state-of-the-art in a wide array of NLP tasks, including translation, question answering, and sentiment analysis.

While attention has been one catalyst of progress in NLP, another was the paradigm shift towards large pre-trained deep neural networks for language processing. From the very beginning of NLP in the 1950s through the 1990s, NLP was done largely by creating collections of rules that attempted to emulate natural language understanding and then applying those rules to whatever text input was given. Starting in the 1990s, machine learning algorithms were applied to NLP tasks in order to use statistical inference that could be trained with large corpora now accessible due to the growth of the web. However, these algorithms still required elaborate feature engineering; these techniques relied on discrete pipelines of separate, intermediate tasks such as part-of-speech tagging and syntactic dependency parsing to feed the machine learning algorithms ("Natural language processing"). The next paradigm shift in NLP was to do away with the classic pipeline altogether and instead use deep neural networks which require no feature engineering. Instead, these deep neural networks are trained on large corpora so that they may learn language representations implicitly from their training data.

What truly ushered in the new age of NLP was the introduction of BERT by Devlin et al.

in 2019. BERT brought together the two ideas discussed above: it was a deep neural network for NLP based solely on attention (like the Transformer), trained on huge text datasets. It achieved then-state-of-the-art performance on eleven NLP tasks. However, BERT's largest contribution to the field of NLP was that its authors published the model. The idea is that BERT has implicitly learned a strong representation of language that can be useful for a wide range of NLP tasks. In order to use BERT for a specific NLP task, researchers only need to build a limited amount of architecture on top of BERT. In this way, BERT acts as a strong basis for any NLP model (and thus it is called a "pre-trained language model", or PLM), and the model is then "fine-tuned" for its specific task. The largest advantage of the PLM/fine-tuning paradigm is that the fine-tuned models do not need large amounts of labeled data for supervised learning. Since BERT already provides the model with a strong implicit understanding of language, the model only needs enough labeled data to learn the specific elements of language relevant to its supervised task. In this manner, PLMs achieve high performance when fine-tuned for supervised tasks, even with relatively little annotated data, and have revolutionized the field of NLP.

Section 1.3 Problem Statement

Given the success of PLMs, there has been a growing interest in investigating what linguistic knowledge they possess implicitly. The body of work that seeks to do so is sometimes called BERTology (Rogers et al., 2020). While there is always interest in interpreting any state-of-the-art model, interpreting PLMs such as BERT and XLNet adds an extra level of curiosity because they are trained solely on unlabeled data; given that these models receive no explicit supervision for things such as syntax and other ideas that we intuitively believe are important for NLP, it is interesting to see if any semblance of such ideas are present in the internal properties of the models. Additionally, there is a large amount of interest in understanding how the internal properties of the PLMs change over fine-tuning tasks.

Researchers have attempted to "crack open" the black-boxes that are PLMs through a number of avenues. Some analyses focus on how the model responds to formulated sentences that are intended to instantiate specific syntactic structures (Goldberg, 2019). Others try to see what knowledge is possessed in the internal vector representations of the models' hidden layers with techniques such as structural probes (Hewitt and Manning, 2019). However, the avenue through which much of the existing literature attempts to interpret these models is the attention mechanism itself. Attention is naturally interpretable and seems to be the primary source of recent NLP performance improvement, and as such can offer meaningful interpretations of the linguistic knowledge possessed by PLMs.

Since BERT was the first PLM, most of the analysis focused on interpreting attention-based models has been conducted on it. However, since its conception in 2019, BERT has already been surpassed by a number of other PLMs, one of which is called XLNet. XLNet improves upon BERT by reintroducing a recurrence mechanism and using a different unsupervised training task. However, XLNet's attention has not yet been analyzed to the degree of BERT's. To begin interpreting XLNet and to see how its inner workings differ from those of BERT, this paper does a wide variety of attention-focused interpretability analyses on both BERT and XLNet. As such, it also serves as a comprehensive summary of much of the current field of BERTology. As an added dimension to the paper, XLNet was fine-tuned for a Twitter-hate-speech-spreader detection task, and the same analyses are applied to this fine-tuned version. Comparing the results of the analyses for XLNet and its fine-tuned version gives insight as to how the PLM changes over fine-tuning.

Chapter 2

Attention-Based Models

Section 2.1

The Transformer

2.1.1. Architecture

As mentioned, the Transformer was the first attention-only model, suggested by Vaswani et al. in 2017. The Transformer model has an encoder-decoder structure, with the encoder on the left of Figure 2.1 and the decoder on the right. The job of the encoder is to take as input as sequence of symbol representations $\mathbf{x} = (x_1, ..., x_n)$ and output a sequence of continuous representations $\mathbf{z} = (z_1, ..., z_m)$. Given $\mathbf{z}$, the decoder then outputs a sequence $\mathbf{y} = (y_1, ..., y_m)$ one element at a time. In order to generate each element of $\mathbf{y}$, the decoder does so in an auto-regressive fashion, meaning it takes as an additional input the previously generated symbols.

There are a lot of aspects of this model, such as the input/output embeddings and positional encodings, that are important to the model but not specifically to analyses in this paper. So for the sake of keeping the content of this paper focused, only the most relevant aspects will be discussed.

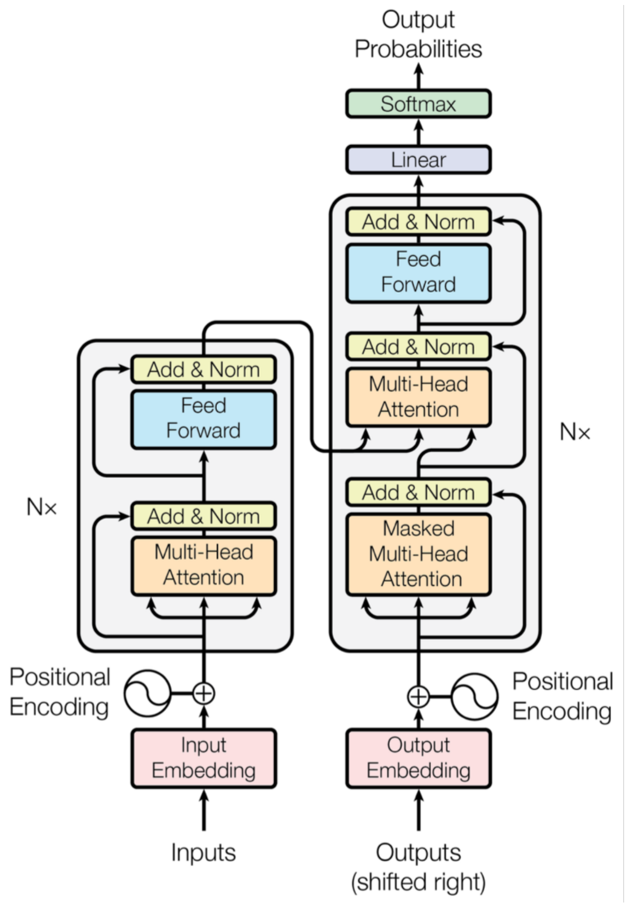

Figure 2.1: The Transformer's Model Architecture

The Encoder Stack. The encoder stack made up of $L = 6$ identical layers. Within each layer, there is a multi-head self-attention sub-layer consisting of a multi-head attention layer, a residual connection, and a layer normalization. The output of that sub-layer is passed to a second sub-layer consisting of a position-wise fully connected feed-forward network with two linear transformations separated by a ReLU activation, also with a residual connection and layer normalization. In order to facilitate the residual connections, all sub-layers in the model and embedding layers produce outputs of dimension $d_{model} = 512$.

The Decoder Stack. The decoder stack is also made up of $L = 6$ identical layers. Each layer of the decoder is similar to the layers of the encoder, with two important differences. The first is that there is an additional self-attention sub-layer in the decoder stack the performs multi-head attention over the output of the encoder stack. The second is that the self-attention sub-layer of the decoder stack is modified to include a mask. This mask helps to ensure that the predictions for position i of the input sequence to the layer can only depend on the known outputs at positions less than i to preserve the sequentiality of the input.

2.1.2. Attention

Scaled Dot-Product Attention. An attention function in general is a function that maps a query and a set of key-value pairs to an output; the output is a weighted sum of the values, with the weight assigned to each value is determined by a compatibility function between the query and the value's corresponding key. There are a number of ways to implement an attention function, but for the Transformer each of the attention sub-layers within the encoder and decoder stacks uses what Vaswani et al. call "Scaled Dot-Product Attention."

Scaled dot-product attention sub-layer functions as such: the function takes in input vectors of dimension $d_{model} = 512$. Call these input vectors $\mathbf{x_1}, ..., \mathbf{x_m}$. For the first layer of the encoder, these inputs are the input embeddings of the original input sequence of symbols. For subsequent layers of the encoder, these inputs are the outputs of the previous layer's feed-forward sub-layer. For each of these input vectors, three new vectors are generated: a query vector, a key vector, and a value vector. The queries and keys are of dimension d_k and the value vector is of dimension d_v. These vectors are generated by multiplying each input vector by learned matrices W^Q, W^K, and W^V, respectively. So for input i there would be generated query, key, and value vectors $\mathbf{q_i} = \mathbf{x_i}W^Q$, $\mathbf{k_i} = \mathbf{x_i}W^K$, and $\mathbf{v_i} = \mathbf{x_i}W^V$. In this case, we are speaking about *self*-attention since the same input vector $\mathbf{x}_i$ is used to calculate both the query vector and the key-value pairings. In this way, self-attention is relating different positions of the same sequence in order to compute the next representation of the sequence (rather than calculating compatibility to some other sequence).

The next step is to calculate how compatible each query vector is with all keys. In scaled dot-product attention, this is done by computing the dot product between $\mathbf{q_i}$ and all keys $\mathbf{k_j} \forall j \in [1, m]$, hence the name. Each of these dot products are scaled by $1/\sqrt{d_k}$ and passed through a softmax operation to compute an attention score between input i and all the other inputs (including itself). Let's call the vector of all these resulting scaled dot-products $\mathbf{a_i}$. The division by $\sqrt{d_k}$ is to counteract large dot-product values that result from a high d_k, which pushes the softmax function into regions where it has extremely small gradients. Thus, this scaling helps keep gradients within an effective range.

The final step of scaled dot-product attention for input i is to weigh the value vector $\mathbf{v_j}$ for each input $j \in [1, m]$ by the attention score computed between input i and j, and then to sum up these scaled value vectors into a single vector, $\mathbf{z_i}$. This $\mathbf{z_i}$ is the next representation of input i that is computed by the self-attention sub-layer.

In practice, the calculations for each of the input vectors $\mathbf{x_1}, ..., \mathbf{x_m}$ are done simultaneously through matrix multiplication. To do so, the input vectors are stacked to form a matrix X of dimensions $m \times d_{model}$. The matrices of all the query, key, and value vectors are calculated by the matrix multiplications XW^Q, XW^K, and XW^V, respectively. Then an attention matrix A can be computed as softmax $\left(QK^T/\sqrt{d_k}\right)$. This attention matrix A can then be multiplied by V to get matrix Z, which is the matrix of the new representations for the inputs in X.

In sum, given a matrix of m inputs $X \in \mathbb{R}^{m \times d_{model}}$, a query weight matrix $W^Q \in \mathbb{R}^{d_{model} \times d_k}$, a key weight matrix $W^K \in \mathbb{R}^{d_{model} \times d_k}$, and a value weight matrix $W^V \in \mathbb{R}^{d_{model} \times d_v}$, then the output Z of scaled dot-product self attention can be computed as such:

$$Q = XW^Q, K = XW^K, V = XW^V \tag{2.1}$$

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{2.2}$$

$$Z = AV = \text{Attention}(X, W^Q, W^K, W^V) \tag{2.3}$$

where the softmax operation is applied to each row of the matrix that results from $QK^T/\sqrt{d_k}$ to produce A.

Before moving forward, it is very important to understand intuitively the properties of the matrix A. Row i of A is the vector $\mathbf{a_i}$ discussed prior to the matrix implementation of scaled dot-product attention. That means that entry $A_{i,j}$ is the weight that input i puts on the value vector of input j when calculating its next representation. Further, since each $\mathbf{a_i}$ has been put through a softmax operation and its entries add up to 1, $A_{i,j}$ can be even more specifically viewed as the *percentage* of input i's total attention that it puts on input j. This is why attention is naturally so interpretable and why it is such a useful tool for understanding the inner-workings of attention-based models; each attention weight has a clear meaning in that it determines how important it is to consider every other token when producing the next representation for the current token. For a given self-attention mechanism, one only needs to look at the distributions of attentions between inputs to gain insight as to what relationships are being drawn between different parts of a sequence.

Multi-head Attention. Rather than have a single instance of scaled dot-product attention (often called an "attention head") in every attention sub-layer, the Transformer combines multiple instances into a single multi-head attention mechanism. This mechanism has H attention heads, where $H = 8$ in the case of the Transformer. Each head h has its own learned query, key, and value weight matrices W_h^Q, W_h^K, and W_h^V, and each produces an output Z_h for the given input X. The output for the entire multi-head attention mechanism, Z, comes from concatenating the outputs from the individual attention heads $Z_1, ..., Z_H$ and then linearly projected by being multiplied by a matrix W^M.

In sum,

$$\text{MultiHead}(X) = \text{Concat}(Z_1, ..., Z_H)W^M \tag{2.4}$$

where

$$Z_h = \text{Attention}(X, W_h^Q, W_h^K, W_h^V) \tag{2.5}$$

Same as before, $W_h^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_h^K \in \mathbb{R}^{d_{model} \times d_k}$, and $W_h^V \in \mathbb{R}^{d_{model} \times d_v}$. Additionally, $W^M \in \mathbb{R}^{Hd_v \times d_{model}}$.

2.1.3. Attention Notation

With attention now being well-defined, it is useful to quickly define the notation that will be used hereafter for referring to attention heads and weights. First, a model has L layers, and H attention-heads per layer. For a given head $h \in [1, H]$ in layer $\ell \in [1, L]$, that head is called ℓ-h. The attention-matrix that is calculated when head ℓ-h processes a sequence of inputs $\mathbf{x}$ is denoted as $\alpha^{\ell,h}(\mathbf{x})$. This was referred to as A in the discussion of scaled dot-product attention. The attention weight that the ith entry of $\mathbf{x}$ puts on the jth entry of $\mathbf{x}$ for the given head is $\alpha^{\ell,h}(\mathbf{x})_{i,j}$.

Section 2.2

BERT

It is worth noting that the Transformer is a general architecture for any sequence processing task. One such class of sequence processing tasks is natural language processing (NLP) since words are sequential in nature, and attention-based models have seen great success in this field. So while it is possible to study attention-based models applied to a wide variety of tasks, this paper focuses on models used for NLP, namely BERT and XLNet.

2.2.1. Architecture

BERT is a multi-layer model that produces **B**idirectional **E**ncoder **R**epresentations from **T**ransformers.[1] The "ERT" part means that it's architecture is just the encoder stack

[1]It seems that researchers in the field of NLP models enjoy naming their contributions after Seasame Street characters. Other contributions besides BERT include ERNIE, ELMo, and BigBird.

of the Transformer as described in §2.1.1. The main difference between BERT and the Transformer's encoder stack is the number of layers L, the number of self-attention heads per layer H, and the dimensionality of the model's hidden representations d_{model}. For the Transformer, $L = 6, H = 8$, and $d_{model} = 512$. BERT has two model sizes: for $\text{BERT}_{\text{BASE}}$, $L = 12, H = 12$, and $d_{model} = 768$; for $\text{BERT}_{\text{LARGE}}$, $L = 24, H = 16$, and $d_{model} = 1024$. This paper studies $\text{BERT}_{\text{BASE}}$. Another important difference is that since BERT is just an encoder, it does not do any auto-regression like the Transformer's decoder stack to produce its outputs. Instead, BERT produces all of its outputs in one pass of the model, rather than outputting each output individually. As such, BERT is an auto-encoder (AE) rather than being auto-regressive (AR).

2.2.2. Pre-Training Tasks

The "B" part of BERT has to do with the techniques used to pre-train the model. Prior to BERT, previous models, such as the Generative Pre-trained Transformer (Radford et al., 2018), used unidirectional language models to learn general language representations. "Unidirectional" means that there is either a left-to-right (or right-to-left) architecture where every entry in the input sequence can only attend to the tokens that came before (or after) it in the self-attention layers of the Transformer. However, Devlin et al. hypothesized that such unidirectionality could be harmful when trying to fine-tune a PLM for word-level tasks such as translation where it is important to incorporate context from both directions of the input sequence.

Thus, BERT's first pre-training objective eliminates unidirectionality by using a "masked language model" (MLM) pre-training objective, which was inspired by the Cloze task (Taylor, 1953). It is also pre-trained on a next-sentence prediction (NSP) task. These tasks use a corpus which is a combination of BooksCorpus (800M words)(Zhu et al., 2015) and text passages from English Wikipedia (2,500M words)(Wikimedia Foundation). It is important to note that neither of these tasks are supervised, as will be described in more detail. Rather, the model ingests a large amount of text as it attempts unsupervised training tasks. Within the context of this paper, it would therefore be remarkable if robust syntactical knowledge was learned without ever being informed of any syntactical rules.

Masked Language Modeling. In the MLM pre-training objective, which in the literature is also referred to as the Cloze task (Taylor, 1953), some tokens of the input are randomly masked or corrupted, and the objective is to predict the masked token based only on its context, which are the non-masked tokens in the input. Specifically, 15% of tokens are selected to be masked. Of those selected, 80% of the time they are replaced with a `[MASK]` token, 10% of the time they are replaced by a random token, and left unchanged 10% of the time.

Next Sentence Prediction. In order to improve the model for downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) which require understanding relationships between multiple sentences, BERT is given an additional pre-training objective. In the NSP task BERT is given input sentences `A` and `B` and is asked to predict whether sentence `B` follows sentence `A` in the original corpus or not. 50% of the time, `B` actually follows `A`, and 50% of the time it is a random other sentence from the corpus.

Section 2.3

XLNet

2.3.1. Architecture

As discussed, part of the reason BERT outperforms AR models is that the bidirectionality of its encodings allow it to better capture context from the entire input sequence, rather than just from the part of the sequence that has been auto-regressed over so far. However, this advantage comes with three main cons.

1. Although the `[MASK]` token is present during pre-training, it never shows up in actual downstream tasks. As such, there is a pretrain-finetune discrepancy in BERT that is not present in AR models (which do not rely on data corruption for training). This discrepancy comes about because models that rely on noisy data for pre-training are prone to learning representations that are covariant with the noise, but that are not consistent with the input received for fine-tuned tasks (Luo et al., 2020).

2. When BERT attempts to use the context of the unmasked tokens to reconstruct a masked token, it cannot use other masked tokens to do so. Essentially, MLM task assumes that the masked tokens are independent of each other and that each can be reconstructed without needing to know the other masked tokens. This is of course a faulty assumption because it is likely that two masked tokens in the same sentence are correlated, yet BERT cannot learn this correlation.

3. Unlike AR models, BERT's input size is limited. It cannot consider contexts larger than 768 input tokens at a time, while an AR model can theoretically consider context across an indefinitely sized input.

XLNet was created by Yang et al. in 2020 to keep the bidirectionality advantage of autoencoders such as BERT while eliminating said disadvantages by introducing principles of autoregression. Principles of AR are introduced in two ways: through architecture differences and through a different pre-training method which is discussed in §2.3.2.

In terms of architectural differences, XLNet takes ideas from Transformer-XL (Dai et al., 2019). The main idea of Transformer-XL was to build a recurrence mechanism on top of the Transformer. The original Transformer could take inputs up to 512 tokens in length. If it wanted to process an input longer than 512 tokens, it would have to process the first 512 tokens, then shift one token over and process the second through 513th tokens separately to generate the next output. However, this leads to context fragmentation, where a token's attention can only consider tokens within the 512-token segment currently being processed. For Transformer-XL to process inputs longer than 512 tokens, it splits the input up into segments and runs the Transformer architecture on each segment, with the key difference being that the hidden state sequence computed over the previous segment is fixed and cached in order to be used for extended context when processing the next segment. In this way the recurrence mechanism is able to capture extended context over indefinitely long inputs. XLNet similarly has a recurrence mechanism, but instead of processing segments with the Transformer architecture, it processes segments with the same architecture as BERT. Thus for XLNet$_{\text{BASE}}$, $L = 12, H = 12$, and $d_{model} = 768$ and for XLNet$_{\text{LARGE}}$, $L = 24, H = 16$, and $d_{model} = 1024$. This paper studies XLNet$_{\text{BASE}}$.

2.3.2. Pre-Training Task

Permutation Language Modeling. In order to capture bidirectional context within an AR framework (and thus without introducing a `[MASK]` token), XLNet proposed a generalized autoregressive pre-training method which they called the permutation language modeling objective. The idea is this: consider a sequence $\mathbf{x}$ of length n. During pre-training, $\mathbf{x}$ is a sequence built from two sentences from the training corpus (see §3.1.1 for more detail). There are $n!$ possible permutations on which an autoregressive factorization could be performed. By autoregressive factorization, we mean the way that autoregressive models are normally trained: for $i \in [0, n-1]$ they are given the first i elements of a sequence and trained to predict the $(i+1)$th element of the sequence. XLNet trains in the autoregressive fashion. However, every time it goes to train over an input sequence $\mathbf{x}$, instead of being given $\mathbf{x}$, it is given one of the $n!$ permutations of $\mathbf{x}$ at random. That means that every time the model needs to predict an element $x_t \in \mathbf{x}$, it will have been given a different subset of $\mathbf{x}$ (as determined by the permutation) as the run-up to the autoregressive prediction. The part of the permutation seen by the autoregressive factorization before being asked to predict x_t can be thought of as the context, and it will in expectation contain elements of $\mathbf{x}$ that come both before *and* after x_t in $\mathbf{x}$. In this way, the model is able to capture bidirectional context using an AR framework.

2.3.3. Performance Comparison to BERT

Yang et al. compared the performance of XLNet to BERT on a number of NLP datasets. In each case, XLNet's and BERT's architectures were trained with the same data and hyper-parameters for fair comparison. The results show that when trained on the same data with the same training recipes (of course varying in their pre-training tasks), XLNet outperforms BERT on all the considered datasets which span a wide range of NLP tasks. For example, SST-2 is a sentiment classification task, while CoLA concerns a model's ability to judge the grammatical acceptability of a sentence.

Task	BERT$_{\text{LARGE}}$	XLNet$_{\text{LARGE}}$
SQuAD1.1	86.7/92.8	88.2/94.0
SQuAD2.0	82.8/85.5	85.1/87.8
RACE	75.1	77.4
MNLI	87.3	88.4
QNLI	93.0	93.9
QQP	91.4	91.8
RTE	74.0	81.2
SST-2	94.0	94.4
MRPC	88.7	90.0
CoLA	63.7	65.2
STS-B	90.2	91.1

Section 2.4

GROVER

2.4.1. Task Definition and Data

To understand how attention patterns may change when a PLM such as BERT or XLNet is fine-tuned for a downstream task, this paper studies the attentions of a third model. This model is XLNet fine-tuned to detect hate-speech spreaders on Twitter. In the spirit of naming attention-based NLP models after Sesame Street characters, I call this fine-tuned model GROVER, which stands for **G**etting **R**id **O**f **V**illains with **E**ncoder **R**epresentations, and who is also one of my personal favorite characters.

This downstream task is the PAN at CLEF 2021 task "Profiling Hate Speech Spreaders on Twitter." The goal is to take a given Twitter feed of some user, and to discriminate between authors that have shared hate speech from those that have never. The dataset consists of 200 Twitter users, with 100 tweets for each user, and a label for whether or not they are a hate-speech spreader.

2.4.2. Fine-Tuning Results

To fit the architecture of XLNet to this sequence classification task (the sequence being a Twitter feed), a single full-connected layer is added on top of the pooled output of the XLNet architecture. It was fine-tuned over 7 epochs of the dataset, after which it achieved a validation accuracy of 68.33%.

An important note about the way that GROVER was fine-tuned is that although the given input Twitter feeds are longer than 768 elements (which XLNet can handle through autoregression), the best performance was found by training only over inputs of 512 elements. To reduce the input Twitter feeds to 512 elements, each tweet of a user was run through an off-the-shelf hate-speech classifier. Then 80% of the tweets classified as non-hate-speech were removed, and then the first 512 elements of what remained was passed in as input. In this way, GROVER's autoregression mechanism was never triggered and was trained as if it were an autoencoder of the same dimensions as BERT starting with the pretrained attention maps of XLNet.

Chapter 3

Surface-Level Patterns in Attention

Section 3.1 Measuring Patterns in Attention

Generally, the method for detecting a pattern within a given attention head functions as such: First, one must define some measure that computes how present that pattern was in the attention weights for a given attention head and model input. For a given attention head $\alpha^{\ell,h}$ that is attending to a model input $\mathbf{x}$, let this measure of pattern presence be called $f_{\alpha^{\ell,h}}(\mathbf{x})$. Second, one must calculate values of $f_{\alpha^{\ell,h}}(\mathbf{x})$ for very many different $\mathbf{x}$. By doing so and then obtaining an average value for how present a given pattern is in an attention head, one can get a good idea for what patterns are consistently computed by the attention head, and to what extent. To formalize even further, let the corpus of all inputs be called $\mathbb{C}$ and let $F_{\alpha^{\ell,h}}(\mathbb{C})$ be the average $f_{\alpha^{\ell,h}}(\mathbf{x})$ value for all $\mathbf{x} \in \mathbb{C}$. That is,

$$F_{\alpha^{\ell,h}}(\mathbb{C}) = \operatorname*{avg}_{\mathbf{x}\in\mathbb{C}} f_{\alpha^{\ell,h}}(\mathbf{x}) \tag{3.1}$$

While it is impossible to create a corpus of all possible inputs, if a large enough corpus is used that is a generally representative sampling of the inputs the model may expect to receive, then $F_{\alpha^{\ell,h}}(\mathbb{C})$ becomes a good indicator of whether a pattern truly is consistently present in the attention of a given attention head.

3.1.1. Corpus

For the analyses in this paper, unless specified otherwise, the corpus $\mathbb{C}$ used was generated in a similar fashion to that used by Clark et al., 2019. The corpus consists of 1000 random sequences from English Wikipedia. A sequence consists of two consecutive paragraphs from a Wikipedia article. To keep sequences within a fairly representative range of normal language, sequences are no less than 40 elements long and no more than 128 elements long. When passed into BERT, a sequence is inputted as "`[CLS]`<paragraph-1>`[SEP]`<paragraph-2>`[SEP]`"; when passed into XLNet, a sequence is inputted as "<paragraph-1><`sep`> <paragraph-2><`sep`><`cls`>". The average number of elements in a sequence in the corpus is 67.192.

3.1.2. Tokens and Words

Before moving forward, it is important to understand how a numeric model can take text as input. Each PLM has its own tokenization scheme which maps a word to a numeric vector assigned to that word (its token). A text input is passed into the model as a concatenated sequence of all the tokens of the words in the sentence. That is, for a sentence $\mathbf{s} = (word_1, word_2, ..., word_m)$, the input to a model $\mathbf{x} = token(word_1) \oplus token(word_2) \oplus ... \oplus token(word_m)$, where *token* is the function that maps a word to its vector representation for the model and $\oplus$ refers to concatenation.

Sometimes the length of a word's token vector is greater than one. For example, BERT's tokenizer maps the word `[CLS]` to $[101]$, but "don't" is mapped to $[1274, 112, 189]$. Because of this, the tokenized version of a sentence $\mathbf{x}$ can have more elements than the sentence itself. This poses a small obstacle to interpretability since the attention mechanisms in the Transformer-based models give token-to-token attentions, while word-to-word attentions would be far more understandable by a human. In order to deal with this, the analyses in this paper convert token-to-token attentions to word-to-word attentions using the same method used by Clark et al.

To do so, for a word split across multiple tokens, the attentions *to* the words are summed up. That is, the total attention paid to a word is the sum of the attentions paid to its tokens. In an attention matrix $\alpha^{\ell,h}$, this is transformation is done by adding the columns corresponding to tokens of the word. For the attention *from* a split-up word, the mean attention weights are taken over the corresponding tokens. That is, the attention the word pays to other words/tokens is the average attention each of its tokens pay to other words/tokens. This transformation is done by doing element-wise averages across the rows of $\alpha^{\ell,h}$ corresponding to the tokens of the word. These transformations result in a word-to-word attention matrix with the same properties as the token-to-token attention matrix, the most important of which is that the attentions from each word sum to 1.

Section 3.2

Relative Position

One of the more simple patterns that can be detected in self-attention is how much attention a token puts on tokens near it in the input sequence. That is, what is the average attention a token puts towards the token that is k positions away from it? The most common relative positional patterns to check for are attention to the previous token ($k = -1$), the current token ($k = 0$), and the next token ($k = 1$). The average attention an attention head $\alpha^{\ell,h}$ puts on the token k away, $RP_{\alpha^{\ell,h}}(k)$, can be computed as

$$RP_{\alpha^{\ell,h}}(k) = \operatorname*{avg}_{\mathbf{x}\in\mathbb{C}} \left(\frac{1}{|\mathbf{x}| - |k|} \sum_{i=1}^{|\mathbf{x}|} \alpha^{\ell,h}(\mathbf{x})_{i,i+k} \right) \tag{3.2}$$

To simplify notation, assume that if a given index pair i, j is not valid (i.e. outside the bounds of the matrix) then $\alpha^{\ell,h}(\mathbf{x})_{i,j} = 0$. In the function that is being averaged across all inputs in the corpus $\mathbb{C}$, for each token x_i in the sequence $\mathbf{x}$ that has a token k away from it, it adds the percentage attention that the token pays to that token x_{i+k}, which is the value of the attention matrix $\alpha^{\ell,h}(\mathbf{x})_{i,i+k}$. It then takes the average of that sum by dividing by $|\mathbf{x}| - |k|$ because that is how many tokens in $\mathbf{x}$ will have a token k away from it (for example, the first token will not have any tokens before it to pay attention to, nor will the last token have tokens after it to pay attention to).

3.2.1. Results

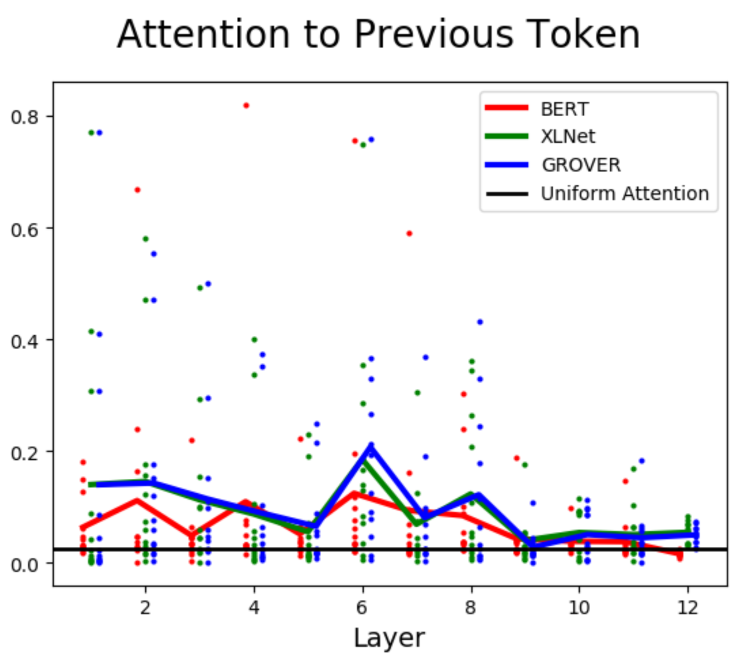

Figure 3.1: Proportion of each head's total attention paid to the previous token in the sequence.

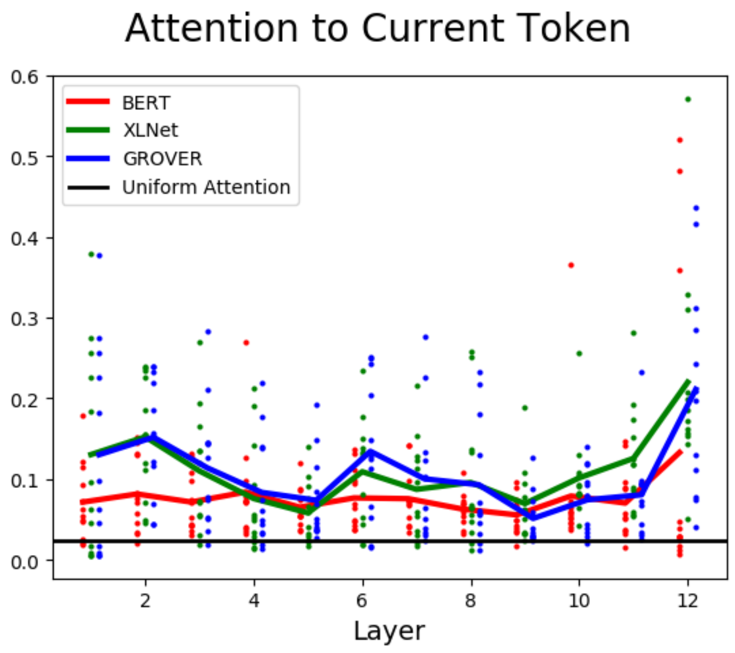

Figure 3.2: Proportion of each head's total attention paid to the current token in the sequence.

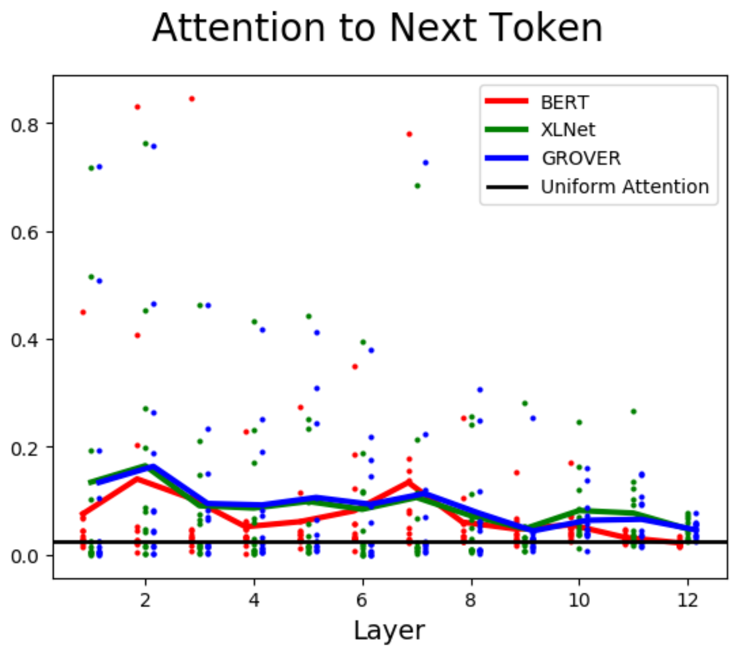

Figure 3.3: Proportion of each head's total attention paid to the next token in the sequence.

As a brief orientation to these graphs: each color represents one of the different models. Each dot on the graph is one head, and the lines represent the average value of the metric for all the heads in a layer. The black uniform line represents what the value for a metric would be if every attention head in the model distributed its attention uniformly across all tokens in the input.

From figures 3.1, 3.2, and 3.3, we see that overall the models pay more attention to the previous, current, and next tokens than would be uniform throughout all layers, but not much more. Many heads in the models pay less attention than would be uniform. However, while overall the models do not place a tremendous amount of importance on these relative positions, there are specific *heads* that do. For example, head 3-4 in BERT places 84.59% of its attention on the next token. One can see that there are similarly specialized heads to current and previous tokens in all three models. Since a uniform attention head would only place 2.43% of its total attention on a given token, significantly different numbers such as 84.59% indicate that that head has specialized in that relationship. In this way, analyzing attention is able to provide insight into what parts of the input the model may be focusing on as part of its analysis.

Section 3.3
Average Distance

Closely related to relative position, it is useful to measure the average number of tokens away an attention head attends to from the current token (Vig and Belinkov, 2019). A positive average distance would indicate that an attention head pays most of its attention to the context of the sentence that comes after the current token, while a negative average distance would mean that the attention head is mainly attending to previous tokens. The the average attention distance of an attention head $\bar{D}_{\alpha^{\ell,h}}$ cab be computed as

$$\bar{D}_{\alpha^{\ell,h}} = \operatorname*{avg}_{\mathbf{x}\in\mathbb{C}} \left(\frac{1}{|\mathbf{x}|} \sum_{i=1}^{|\mathbf{x}|} \sum_{j=1}^{|\mathbf{x}|} \alpha^{\ell,h}(\mathbf{x})_{i,j} \cdot (j - i) \right) \tag{3.3}$$

Here in the inner function, for each token x_i in the sequence, it is calculating a weighted average of its attention to all other tokens in the sequence, weighted by how far away each token is from x_i. The average distances from each token are summed up, and then averaged over the number of tokens in the sequence $|\mathbf{x}|$ to get the average attention distance the head has between input tokens.

3.3.1. Results

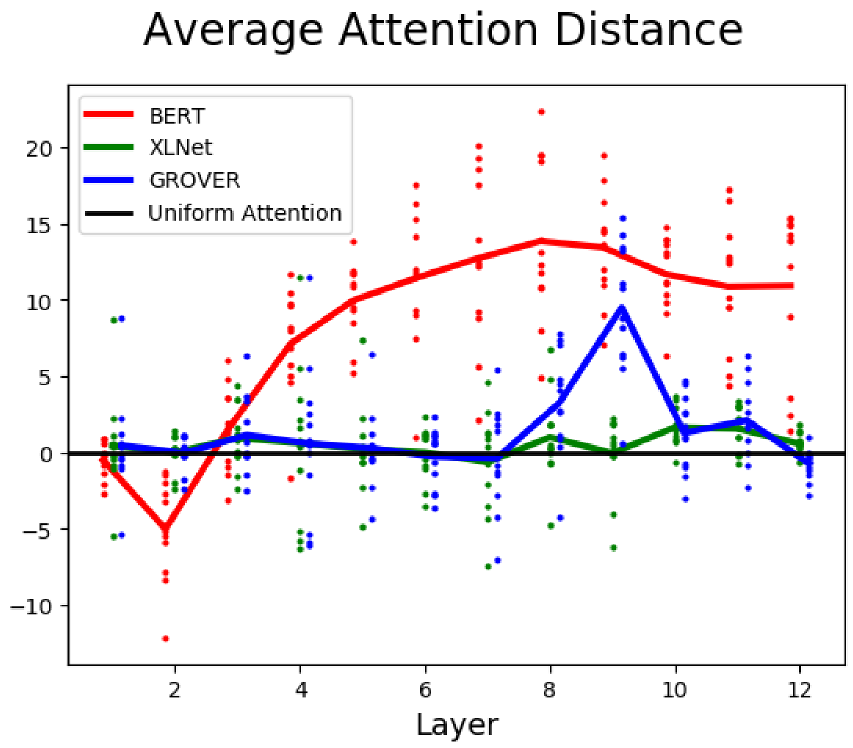

Figure 3.4: Average number of tokens away from the current token that each head attends to.

This graph shows one of the starkest differences between the three models. BERT distributes its attention fairly normally around the current token in the first and third layers, then looks primarily behind the current token in the second layer, and then looks primarily forward in the rest of its layers. After the third layer, no head besides one in the fourth layer in BERT has its average attention look behind the current token. While BERT attention heads are certainly able to capture bidirectional context, the vast majority of attention paid in the backwards direction is paid in the first three layers, and thereafter the majority of attention is paid in the forward direction.

In stark contrast, XLNet does not put more emphasis on either context direction in any one layer, remaining fairly balanced throughout the model. Specific XLNet heads place their average attention in either direction, but as a whole XLNet puts even amounts of attention towards both context directions. In this way, XLNet seems to achieve the goal of being bidirectional more than BERT, which puts more of its attention in the forward direction, despite the "B" in BERT standing for "Bidirectional."

Section 3.4 Entropy

Another informative attention pattern is the entropy of an attention head. Entropy measures whether an attention head attends broadly over many words or focuses on a few, and to what extent it does so. Since the attention weights of all the rows in an attention head add up to 1 (due to the softmax operation), they can also be viewed like a probability distribution of a discrete random variable. In discrete probability, for a discrete random variable Y with possible values $y_1, ..., y_n$ and a probability mass function $P(Y)$, the entropy of Y, $H(Y)$ (Shannon, 1948), can be calculated as

$$H(Y) = -\sum_{i=1}^{n} P(y_i) \log_2 P(y_i) \tag{3.4}$$

This value $H(Y)$ is a measure of the level of uncertainty inherent in the variable's possible outcomes. If a probability distribution is concentrated on one or a few outcomes, then its outcome will tend to be more predictable than a probability distribution distributed evenly across all possible outcomes, in which case one would not be able to predict one outcome being more likely than any other one. As such, entropy is at its maximum value for a uniform probability distribution. Since attention weights are distributed across a discrete set of tokens just as probabilities are distributed across a discrete set of outcomes for a random variable, entropy can also be applied to attention weights as a measure of how distributed attention is. As such, the entropy of an attention head $H_{\alpha^{\ell,h}}$ can be computed as

$$H_{\alpha^{\ell,h}} = \operatorname*{avg}_{\mathbf{x}\in\mathbb{C}} \left(\frac{-1}{|\mathbf{x}|} \sum_{i=1}^{|\mathbf{x}|} \sum_{j=1}^{|\mathbf{x}|} \alpha^{\ell,h}(\mathbf{x})_{i,j} \log \alpha^{\ell,h}(\mathbf{x})_{i,j} \right) \tag{3.5}$$

In the context of self-attention in NLP, a high entropy value would indicate than an attention head generally attends over many of the tokens in the input sequence, while a low entropy would indicate that it focuses its attention on only a few tokens (irrespective of how far away from the current token the attended tokens are).

3.4.1. Results

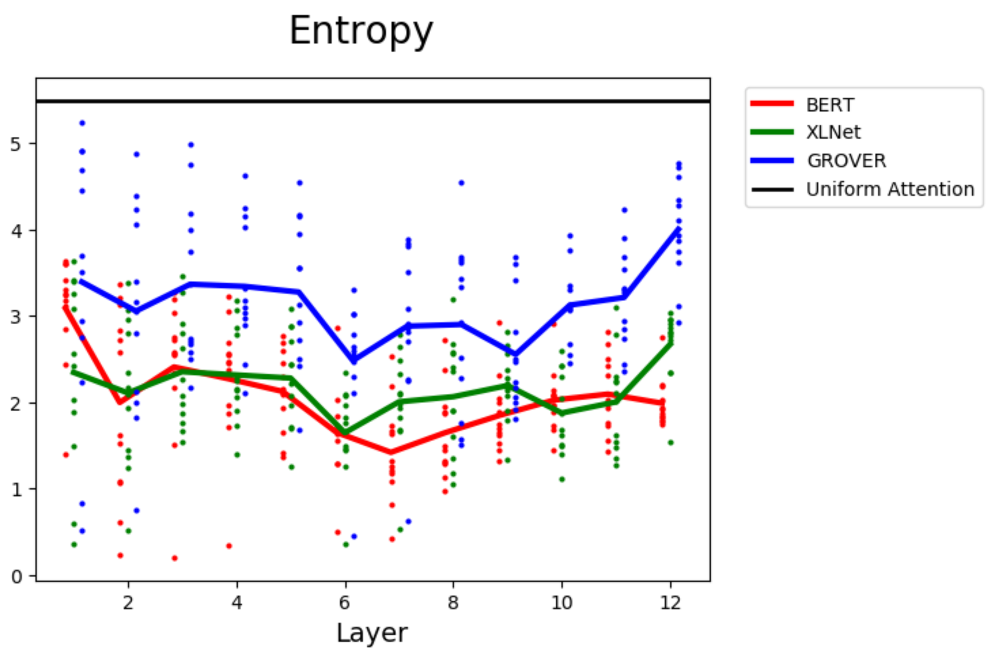

Figure 3.5: Average entropy of each head's attention distribution.

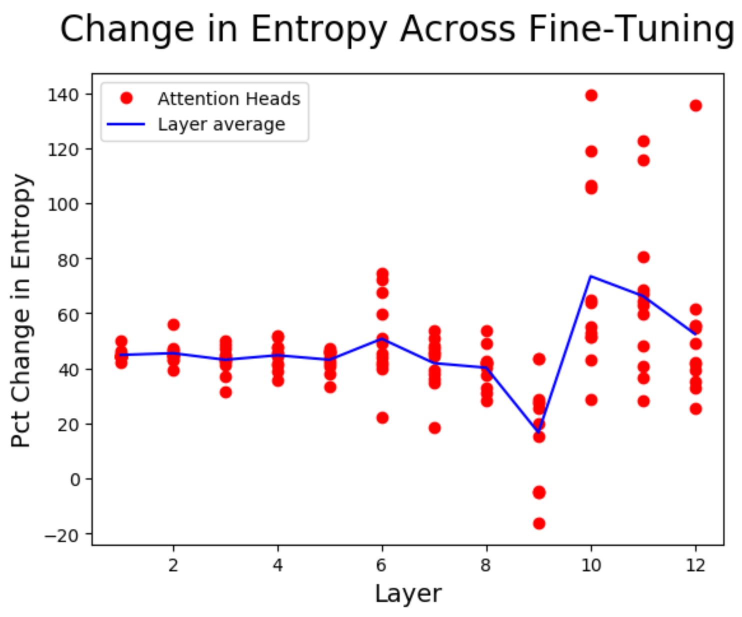

Figure 3.6: Percentage Change in Entropy by Head between XLNet and GROVER.

The pattern of entropy is similar across the models: each attends broadly in early layers, becomes more focused in middle layers, and attends broadly once again in the last layers. Additionally, each model has the head that attends the most broadly in its first layer. Of note, BERT and XLNet have very similar levels of average entropy throughout their layers, so focus of attention does not seem to be a distinguishing factor between the models. However, GROVER's entropy is significantly higher than that of XLNet's, with layers 10, 11, and 12 in particular increasing in entropy, as shown in Figure 3.6. Layer 9 increases in entropy the least relatively, but still increases by 20% on average nevertheless. A possible explanation for this increase in entropy is that GROVER needs to look over an entire Twitter feed to determine if someone is a hate-speech-spreader, which could be determined by only a few tweets out of a hundred. Therefore, it makes sense that GROVER attends broadly over its inputs.

Section 3.5

Variability

It is important to distinguish attention heads that are content-dependent from those that are content-independent (Vig and Belinkov, 2019). Content-independent refers to attention heads that have generally the same patterns present in their attention, regardless of what the actual input sequence is. An example of this would be an attention head that always attends from the current token to the next token. Conversely, a content-dependent head may vary how it distributes its attention depending on the actual input sequence. As such, a measure of variability is a useful tool in discerning which attention heads are content-dependent and which are content-independent, with content-dependent heads having a higher variability than content-independent ones. The attention variability of a given head, $Var_{\alpha^{\ell,h}}$, can be computed as

$$Var_{\alpha^{\ell,h}} = \operatorname*{avg}_{\mathbf{x}\in\mathbb{C}} \left(\sum_{i=1}^{|\mathbf{x}|} \sum_{j=1}^{|\mathbf{x}|} |\alpha^{\ell,h}(\mathbf{x})_{i,j} - \bar{\alpha}^{\ell,h}_{i,j}| \right) \tag{3.6}$$

where $\bar{\alpha}^{\ell,h}_{i,j}$ is the mean of $\alpha^{\ell,h}(\mathbf{x})_{i,j}$ over all $\mathbf{x} \in \mathbb{C}$. Thus, $Var_{\alpha^{\ell,h}}$ is the mean absolute deviation of $\alpha^{\ell,h}$ over $\mathbb{C}$.

3.5.1. Results

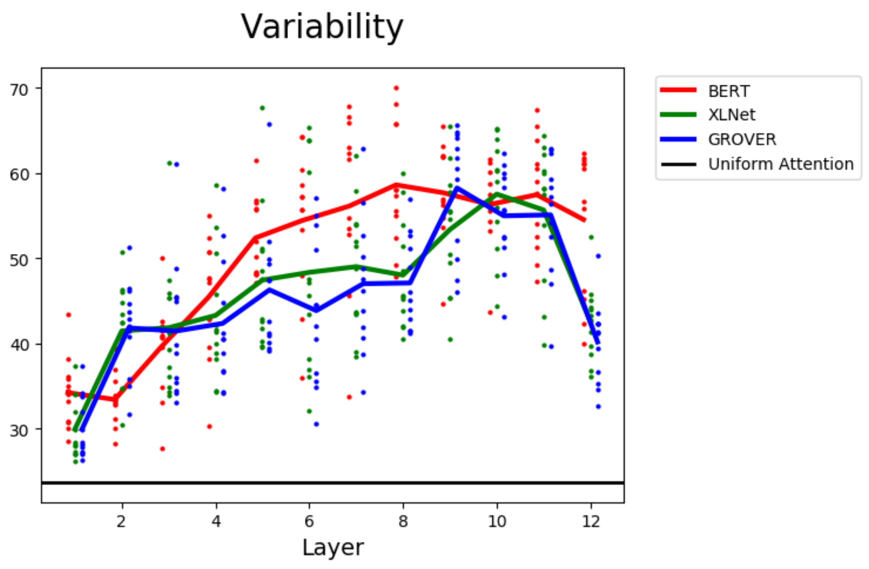

Figure 3.7: Average variability in each attention head's attention weights. The uniform attention model does not have zero variability since inputs have different lengths, and thus different uniform probabilities.

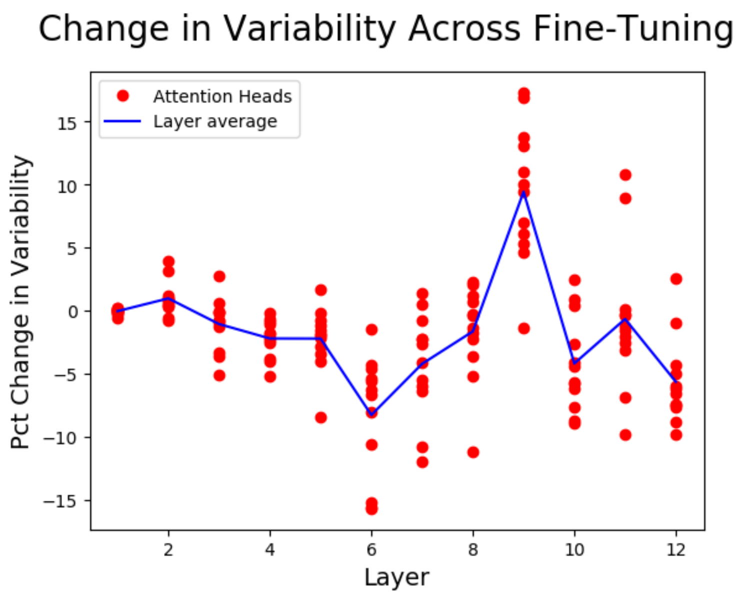

Figure 3.8: Percentage Change in Variability by Head between XLNet and GROVER

Similar to entropy, BERT and XLNet overall seem to have similar patterns of variability, with a few key exceptions. The first is that in the middle layers, BERT has slightly more variability. The second, more prominent difference is that the variability in BERT's last layer is much higher than that of XLNet (and GROVER). If variability is a measure of content-dependence as Vig and Belinkov say (2019), then it would appear that BERT's attention heads tend to be more content-dependent than those of XLNet. Perhaps surprisingly, more layers than not decrease in variability as XLNet is fine-tuned into GROVER. However, as Figure 3.8 shows, there is a large increase is variability in layer 9, so it is not fair to say that GROVER is less content-dependent than XLNet. Additionally, individual heads in every layer of GROVER increase in variability, and GROVER need not increase in variability for every head to introduce specific content-dependence into the model.

Chapter 4

Token-Level Patterns in Attention

Slightly more specific than surface-level patterns are token-level patterns. While surface-level patterns describe how an attention head reacts generally to inputs from the corpus, token-level patterns describe how an attention reacts when specific tokens are passed in as input.

Section 4.1

Token-Specific Patterns

Token-specific patterns measure, for a given token, how much attention on average does the attention head pay towards that token. That is, if a given token is present in an input sequence, what average percentage of its attention does each token in the input sequence put towards the token of interest. If certain heads seem to be specialized in attending to a specific token, this could be an indicator of that token's general importance to the decisions of the model. The average attention an attention head places on a given *token*, $T_{\alpha^{\ell,h}}(token)$, can be computed as

$$T_{\alpha^{\ell,h}}(token) = \operatorname*{avg}_{\mathbf{x}\in\mathbb{C}'} \left(\frac{1}{|\mathbf{x}|} \sum_{i=1}^{|\mathbf{x}|} \sum_{j=1}^{|\mathbf{x}|} \alpha^{\ell,h}(\mathbf{x})_{i,j} \cdot \mathbb{1}_{x_j=token} \right) \tag{4.1}$$

where $\mathbb{1}_{x_j=token}$ is an indicator function which is equal to 1 if the jth token of $\mathbf{x}$ is the token of interest, *token*, and 0 otherwise. Further, $\mathbb{C}'$ is the subset of the corpus $\mathbb{C}$ that contains at least one instance of *token* in the sequence. Without this subsetting, the average attention estimations would be skewed downward, especially for tokens that are less frequent in the corpus. The inner function sums the total attention being paid to instances of *token*, and then averages it by dividing by the total number of tokens in the input sequence.

4.1.1. Results

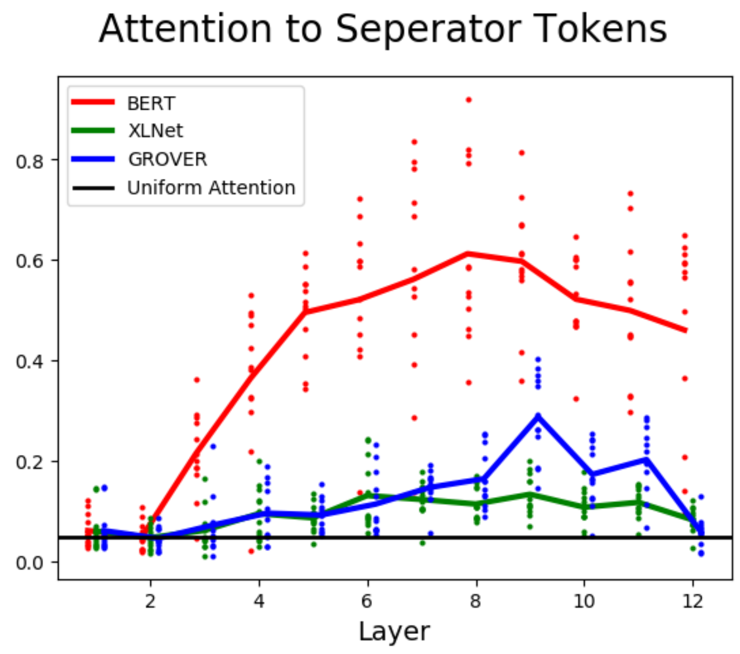

Figure 4.1: Proportion of total attention put towards separator tokens (`[SEP]` in BERT, <`sep`> in XLNet and GROVER).

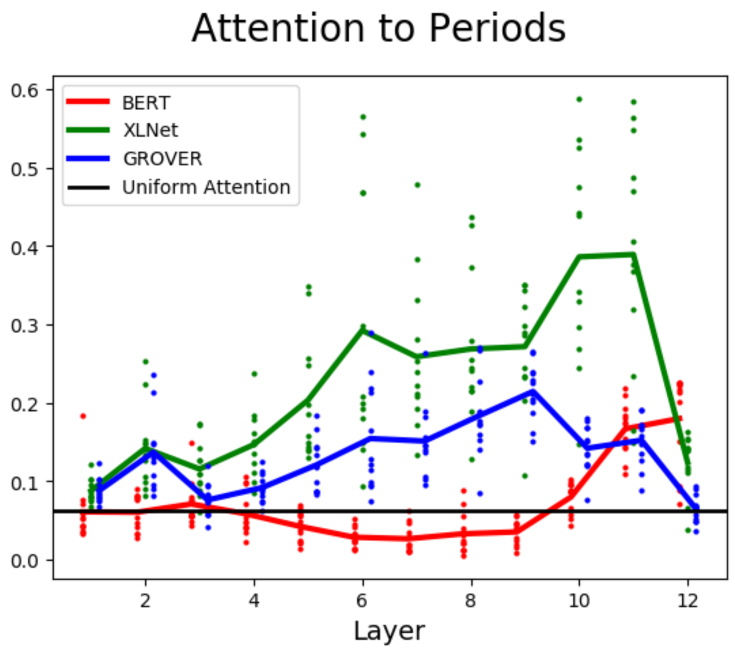

Figure 4.2: Proportion of total attention put towards periods.

The above figures show that some inputs such as special tokens and punctuation are paid far more attention to than they would be in a uniform-attention model. For BERT, it pays a significant amount of attention to `[SEP]` tokens as shown in Figure 4.1, with head 8-10 paying 92.0% of its total attention to them. While XLNet does not pay as much attention to `[SEP]` tokens, it does pay a significant amount of attention to periods as shown in Figure 4.2, with heads 10-5 and 11-2 both putting over 58% of their total attention to them. Given how the input sentences to the models are constructed, as described in §3.1.1, `[SEP]` tokens separate sentences, which end in periods. As such, [SEP] tokens are frequently next to periods, and both indicate the end of sentences, so it appears both models pay a significant amount of attention to the ends of sentences, where BERT "chose" to use `[SEP]` tokens to demarcate the end of sentences and XLNet chose to use periods.

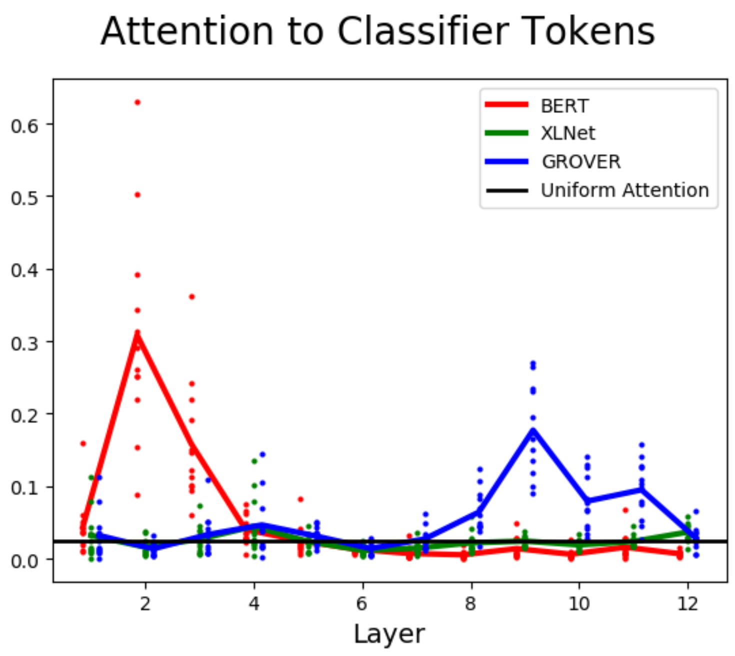

Figure 4.3: Proportion of total attention put towards classification tokens ([CLS] in BERT, <`cls`> in XLNet and GROVER).

Interestingly, XLNet seems to pay attention to classification tokens just as a uniform attention distribution would, while BERT does pay some attention to classification tokens (and has head 2-3 put 63.05% of its total attention to them), but only in the early layers. Layer 4 and onward, BERT similarly pays uniform-like attention to them. Additionally, GROVER has a large increase in attention to classification tokens in later layers, particularly in layer 9, with head 9-4 paying 27.09% of its total to them and heads 9-3, 9-9, and 9-7 all putting over 20% of their attention on them. Given that it is trained for a sentiment classification task, it makes sense that GROVER puts more attention on classification tokens, perhaps as a way to aggregate information relevant to the sentiment classification task that can be used by heads in following layers.

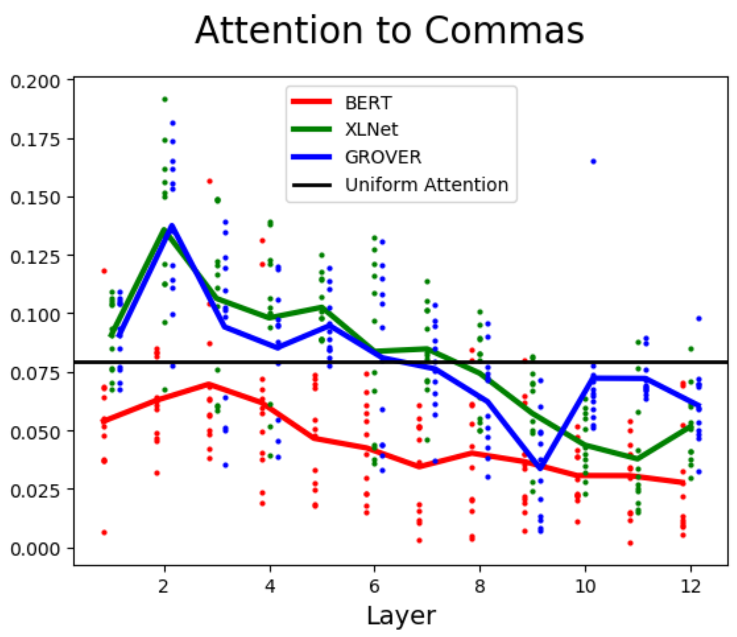

Figure 4.4: Proportion of total attention put towards commas.

Both models pay limited attention to commas as shown by Figure 4.4, with most heads in BERT paying less attention to them than uniform attention and the most attentive head (3-9) paying only 15.67% of its total attention to them. This would suggest that BERT actively down-regulates how much attention is paid to commas and does not use them extensively to extract syntactic or semantic information from its inputs. XLNet pays slightly more attention to commas than uniform attention in early layers, with head 2-10 paying 19.17% of its total attention to them. However, after layer 6 on average XLNet attention heads pay less attention to commas than would be uniform, once again suggesting purposeful down-regulation.

Section 4.2

Token-to-Token Patterns

An even more specific type of token-level pattern is looking at the average attention from a specific token $token_1$ to another specific token $token_2$. The average attention payed from $token_1$ to $token_2$, $T2T_{\alpha^{\ell,h}}(token_1, token_2)$, can be computed as

$$T2T_{\alpha^{\ell,h}}(token_1, token_2) = \operatorname*{avg}_{\mathbf{x} \in \mathbb{C}'} \left(\frac{\sum_{i=1}^{|\mathbf{x}|} \sum_{j=1}^{|\mathbf{x}|} \alpha^{\ell,h}(\mathbf{x}) \cdot \mathbb{1}_{x_i = token_1} \cdot \mathbb{1}_{x_j = token_2}}{\sum_{i=1}^{|\mathbf{x}|} \mathbb{1}_{x_i = token_1}} \right) \tag{4.2}$$

where $\mathbb{C}'$ is the subset of the corpus $\mathbb{C}$ that contains at least one instance of both of $token_1$ and $token_2$. Rather than dividing by $|\mathbf{x}|$ as in $T_{\alpha^{\ell,h}}(token)$, we need to divide only by the number of instances of $token_1$ in the sequence, which is the number calculated in the denominator of the inner function.

4.2.1. Results

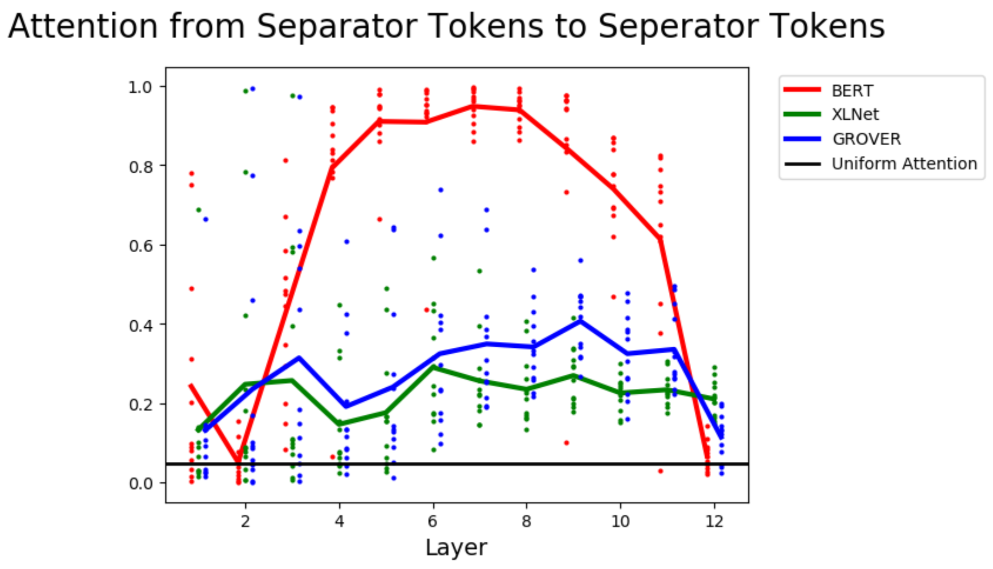

Figure 4.5: Proportion of total attention that separator tokens put towards themselves or other separator tokens.

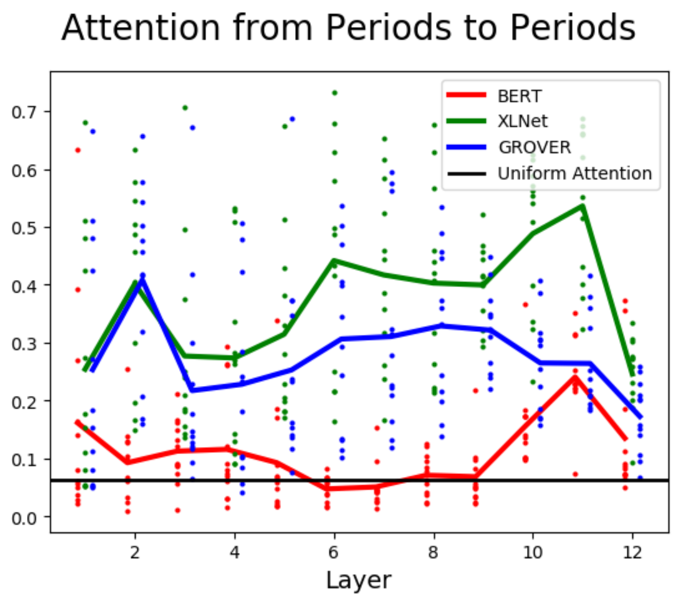

Figure 4.6: Proportion of total attention that periods put towards themselves or other periods.

As shown in Figures 4.1 and 4.2, BERT and XLNet put a lot of attention towards `[SEP]` tokens and periods, respectively. As Clark et al. (2019) mentioned, one hypothesis could be that the high amount of attention paid to these tokens would be to aggregate segment-level info that could be used by other heads. However, if this was the case then one might expect to see these tokens attend broadly over the whole segment, but this does not seem to be so, as illustrated by Figure 4.5. Especially in the middle layers, BERT attention heads processing `[SEP]` tokens put almost all of their attention on other [SEP] tokens. A similar pattern (though to a lesser extent) is found with periods in XLNet, as shown by Figure 4.6, furthering the idea that XLNet treats periods like BERT treats `[SEP]` tokens. As a potential explanation for this phenomenon, Clark et al. (2019) did some qualitative analysis which suggests that when an attention head attends to a specific function, such as identifying dependency relations as discussed in §5.3, it attends to separator tokens (or periods for XLNet) when that function is not present in the input. In this way, separator tokens or periods may be used as a sort of "no-op" when an attention head's function is not applicable.

Section 4.3

Specialization from Fine-Tuning

4.3.1. Results

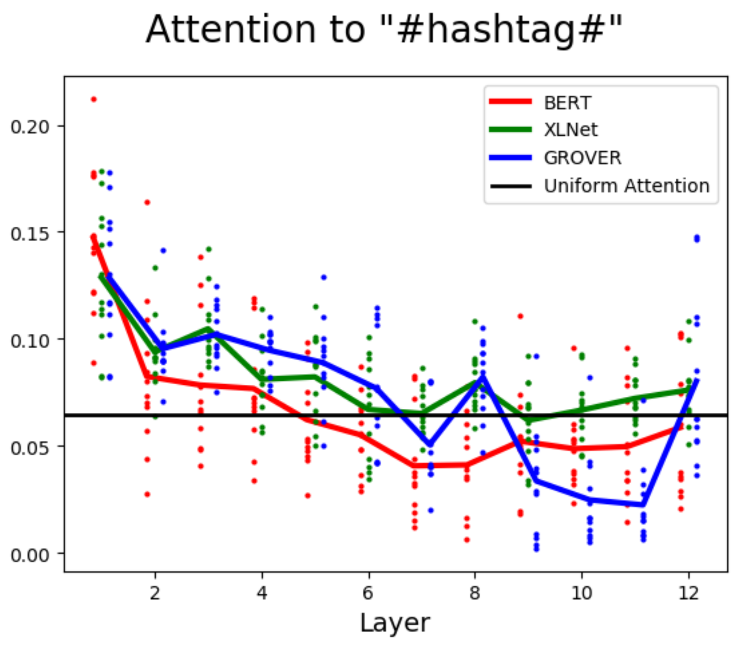

Figure 4.7: Proportion of total attention put towards "#hashtag#".

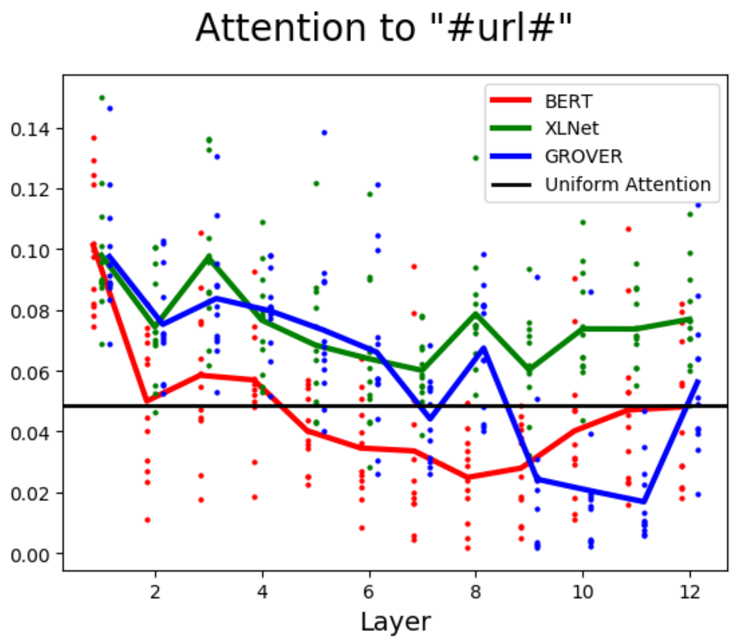

Figure 4.8: Proportion of total attention put towards "#url#".

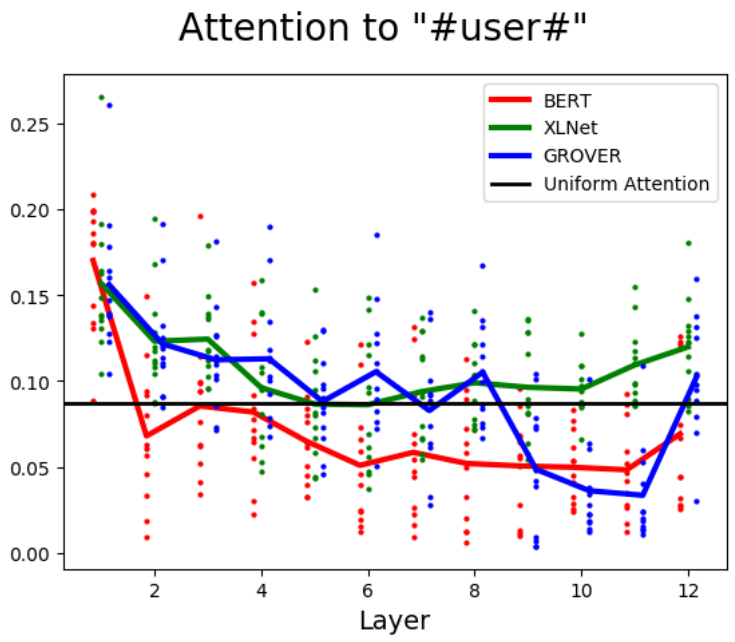

Figure 4.9: Proportion of total attention put towards "#user#".

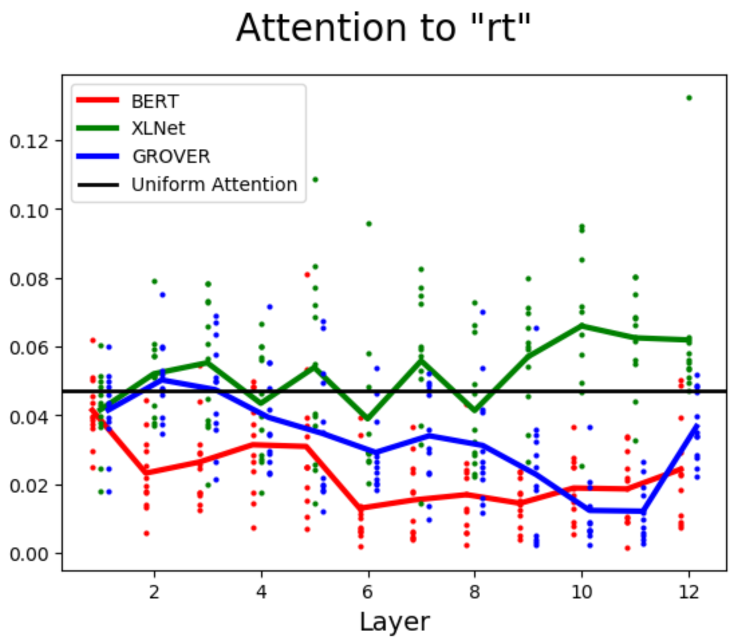

Figure 4.10: Proportion of total attention put towards “rt”.

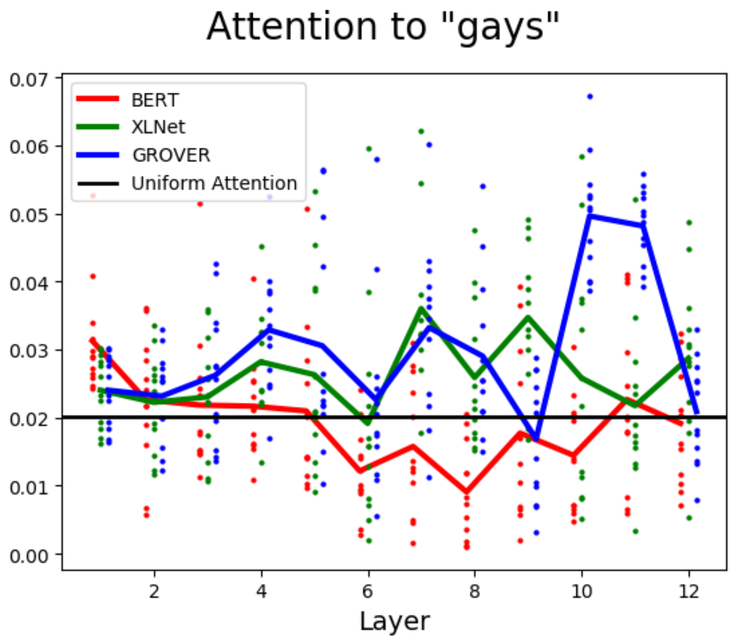

Figure 4.11: Proportion of total attention put towards “gays”.

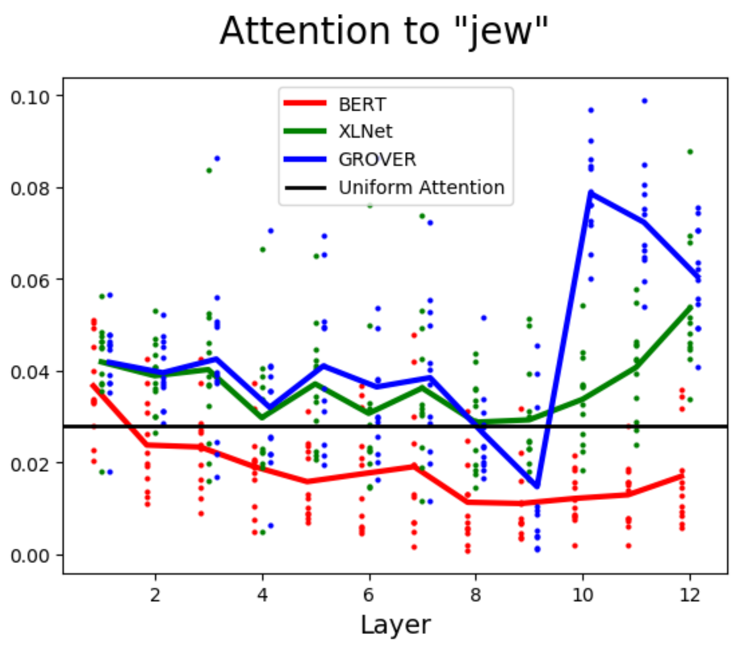

Figure 4.12: Proportion of total attention put towards “jew”.

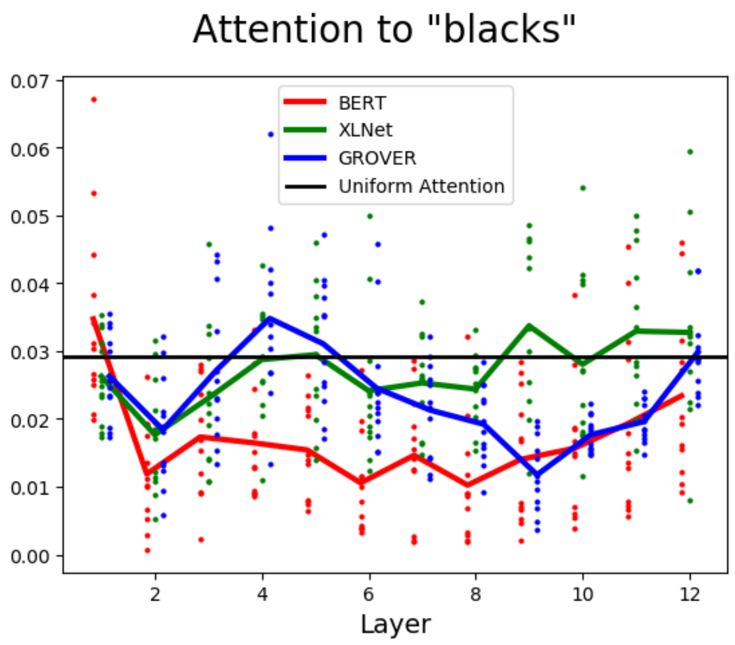

Figure 4.13: Proportion of total attention put towards “blacks”.

Word	Hate-Speech	Non-Hate-Speech
#hashtag#	3392	3757
#url#	6768	8571
#user#	17661	17356
rt	6705	8285
gays	8	2
jew	116	19
blacks	41	6

Figure 4.14: Frequency of words of interest in the the Twitter feeds of authors labeled hate-speech-spreaders and non-hate-speech-spreaders in the dataset GROVER was trained on.

Above are the attentions paid to a number of hand-selected input words specific to the domain of the hate-speech-spreader detection task. The tweets given in the dataset are partially cleaned, with specific hashtags, URLs, and user-mentions replaced with "#hashtag#," "#url#," and "#user#," respectively. Additionally, a tweet that is a re-tweet starts with "rt." The words "gays," "jew," and "blacks" were selected based on qualitative analysis reading a random subset of the dataset and finding topics that seem to be frequently mentioned by hate-speech spreaders.

For the domain-specific tokens that do not carry any semantic meaning (i.e. '#hashtag#," "#url#," "#user#," and "rt"), the patterns seem to be the same: XLNet pays slightly more than uniform attention to these tokens, and GROVER pays a similar level of attention to these tokens in early layers, and then in later layers, usually starting in layer 9, starts to pay less attention to these tokens than XLNet. Given their similar level of usage between hate-speech-spreaders and non-hate-speech-spreaders, it makes sense that GROVER decrease how much attention it pays to these tokens, as they do not seem to be necessarily useful for making classification decisions between the two. Further, this shows that just because a token is frequent in a dataset, such as "#user#," it does not necessarily mean that it will receive a proportional amount of attention. This provides additional color to the discussion about why BERT and XLNet pay so much attention to `[SEP]` tokens and periods, respectively: it is not just because they are frequent in the data, so they must provide some function, perhaps as a "no-op" suggested by Clark et al. (2019).

Conversely, it does seem that words such as "gays" and "jew" provide some function to GROVER, given their imbalance in frequency between hate-speech-spreader tweets and non-hate-speech-spreader tweets. As such, attention heads in later layers, especially layers 10 and 11, increase significantly their attention to these words. The large attention increase to "gays" is particularly interesting because it not a frequent word altogether, with only 10 instances in the dataset (2 of them belonging to non-hate-speech-spreaders), yet GROVER still almost triples the amount of attention layers 10 and 11 pay to it. In this way, changes in attention give us indication as to what specific features of inputs are particularly important for their downstream task. However, one must be hesitant to think that changes in attention patterns are an infallible way to discern what is important about an input for determining the correct output. As a contrast to "gays" and "jew," the word "blacks" is similarly imbalanced between use in hate-speech tweets and non-hate-speech tweets, yet GROVER actually places less attention on this word in later layers than both XLNet and uniform attention, as shown in Figure 4.13. However, it does place more attention on it in layer 4 than XLNet does, with one head (4-3) increasing significantly how much attention it places on the word, so the lack of an overall attention spike also does not infallibly deny that the word is important to the model's decisions. In general, changes in token-level attentions can be used to indicate which tokens may be important for a fine-tuned model's downstream task, but it cannot in itself completely confirm nor deny an token's overall importance to the model's outputs.

Chapter 5

Probing

Section 5.1

Method

The general idea behind probing attention heads is to see if their frozen representations can be used for a supervised task with minimal additional information or variables. The idea is that if attention maps can be used simply but to good effect, then it is likely that information relevant to the supervised task is present within the attention of the model.

The simplest type of probe is that which only considers a single attention head. To probe a single attention head, one treats it as a simple no-training-required classifier: an input is passed into the model, the attention weights for a head is extracted, and whichever word is most attended to by the attention head is treated as its "prediction". Through this process, an attention head's prediction accuracy can be estimated; if its accuracy is high, then that attention head can be considered to at least in part encode the information being tested. Single-headed probes are used to check for part-of-speech and syntactical dependency knowledge in §5.2 and §5.3.

A more complex probe involves multiple heads and may be used to test whether knowledge is possessed within, but possibly distributed across, the overall attention of a model. To do so, an input is passed into the model, the attention weights are extracted and then combined through a simple weighted linear function. The weights are trained through standard supervised training, with the attention heads staying fixed. The idea is that if the trained linear model can perform well on a nonlinear task, then it is likely that knowledge relevant to the nonlinear task is encoded in the attention heads themselves. A multi-head probe is used to induce a dependency tree structure of a input sequence in §5.4.

5.1.1. Corpus

In order to probe for a supervised task, an annotated dataset is needed. The probes in this paper are concerned with probing for different linguistic features, namely parts-of-speech and dependency syntax. In order to obtain a corpus annotated with these linguistic features, this paper uses the same Wikipedia-based corpus as the previous analyses, but annotates it automatically using the Stanza NLP package (Qi et al., 2020). The pipeline used to annotate the data is fairly accurate at annotating the linguistic features of concern: on the GUM universal dependencies treebank, it tags parts of speech with 95.89% accuracy, identifies unlabeled dependency relationships with 87.06% accuracy, and correctly labels 83.57% of dependency relationships.

Section 5.2

Part-of-Speech Recognition

As described in §5.1, individual attention heads can be used as a simple classifier for a supervised task to see if they possess knowledge relevant to that task. One such task is part-of-speech tagging. In order to do so, for each part-of-speech, all inputs from the corpus are passed into each model and the attention weights are extracted. For all input sentences with at least one instance of that part-of-speech, it is measured how frequently the most-attended-to token in each attention head is that part-of-speech. This frequency is considered the head's accuracy at the part-of-speech recognition task.

5.2.1. Results

	BERT		**XLnet**		**GROVER**	
Part of Speech	**Head**	**Acc**	**Head**	**Acc**	**Head**	**Acc**
VERB	1-8	39.48	12-3	39.34	2-12	38.78
NUM	12-3	41.53	3-10	43.72	12-5†	43.99
CCONJ	1-8	24.48	5-10	27.93	1-10†	27.24
PUNCT	1-12	56.24	1-1*	57.63	1-9†	54.62
AUX	11-12	29.17	12-12	30.73	8-9	34.86
ADJ	3-12	33.81	12-2	34.19	8-9	35.90
PRON	1-6	25.94	1-8	29.58	5-6	29.12
DET	1-10	39.31	1-9*	41.58	5-6†	42.63
PROPN	2-1	80.68	5-5	73.38	12-10†	74.42
NOUN	1-5	50.35	12-12	55.63	7-11	55.41
ADV	1-10	26.97	1-1	24.72	2-12	24.72
ADP	1-10	45.43	5-8	50.11	11-3	48.87

Figure 5.1: Top performing heads for part-of-speech recognition. * indicates that the top performing head for XLNet was also present in the top 5 performing heads for GROVER. † indicates that the top performing head for GROVER was also present in the top 5 performing heads for XLNet.

As is seen in Figure 5.1, there are attention heads in each of the three models that do remarkably well at attending to specific parts of speech. For example, 80.68% of the time, BERT's head 2-1's most-attended-to token is a proper noun (if one is present in the sentence). There are similarly performant heads in XLNet and GROVER. Other parts-of-speech that are of interest are how certain heads in each of the models do particularly well at attending to verbs, numbers, punctuation, and nouns. This is extremely interesting to see given the unsupervised nature of these models' pre-training tasks, since they were never supervised in order to identify any part of speech.

In terms of differences between the models, there is not a significant difference in performance across the models for any given part-of-speech: the best head in each model is roughly comparable, and many times are extremely similar. Therefore, it does not seem that any model has distinctly more knowledge regarding part-of-speech tagging than any of the others. Between XLNet and GROVER, the accuracy of the best performing attention heads are very similar. However, it is interesting to see that for no part-of-speech is XLNet's best performing head also GROVER's best performing head. Additionally, it is rare that XLNet's best performing head remains in the top-five performing heads in GROVER, and similarly it is not frequent that GROVER's best performing head was in XLNet's top-five.

Section 5.3 Labeled Dependency Recognition

In linguistics, dependency is the idea that words are connected to each other by direct links, with one word being the root of a sentence and all other words being connected to one other word (its syntactical head or "parent") by some specified relationship. Thus, all words in a sentence are connected to each other either directly or indirectly. Some types of direct relationships between words include connections between nouns and words that modify them, connections between possessive pronouns and the subject they correspond to, and connections between direct objects and their verb.

So to probe single attention heads for dependency knowledge, an input sequence with a specific dependency relationship present is passed into the entire model, the attention heads are extracted, and each head "predicts" the syntactical parent for a given token to be the token most attended to by the attention head for that given token. One can measure how accurate a given attention head probe is at predicting the syntactical parent of a given token generally (unlabeled dependency), or at predicting the syntactical parent of a given token for a specific type of syntactical dependency (labeled dependency). If a single-head probe shows a high accuracy in either the unlabeled or labeled syntactical-parent-recognition task, then it would suggest that that dependency relationship is attended to by the attention head.

In order to establish a baseline to compare the single-head probes to, for each dependency relation a fixed offset is used. For example, for the “obj” relation (verb pointing the the object it corresponds to) an offset of -2 is used, meaning to just predict the word that comes 2 before the given verb. For each dependency relationship, each possible offset between -4 and 4 is tried and the best performing offset is used as the baseline.

As in Figure 5.1, * indicates that the top performing head for XLNet was also present in the top 5 performing heads for GROVER and † indicates that the top performing head for GROVER was also present in the top 5 performing heads for XLNet.

5.3.1. Results

	Baseline		**BERT**		**XLnet**		**GROVER**	
Dependency	**Offset**	**LAS**	**Head**	**LAS**	**Head**	**LAS**	**Head**	**LAS**
UNLABELED	1	26.78	2-3	25.12	2-8*	28.26	2-8†	29.67
nummod	-1	39.67	1-4	43.44	3-9*	49.76	3-9†	52.57
nmod	-2	30.54	1-2	21.58	3-3*	28.16	4-10†	30.32
advmod	1	52.32	7-10	51.77	2-6*	51.84	4-5†	56.07
flat	-1	68.75	6-8	32.62	2-9*	58.72	2-9†	58.47
aux	1	80.47	6-7	87.91	4-5*	90.93	4-5†	90.98
cc	1	52.47	7-5	63.21	3-9*	62.46	3-9†	63.93
compound	1	67.98	7-10	41.72	3-6*	59.89	3-6†	60.05
obj	-2	33.05	6-8	60.61	3-3*	68.63	3-3†	66.40
acl	-1	28.89	7-6	28.46	3-4*	29.37	6-1	31.04
amod	1	68.44	7-10	60.98	1-6*	63.58	1-6†	63.58
det	1	47.89	7-5	88.13	3-9*	82.22	3-9†	82.77
nsubj	1	26.93	7-10	22.36	3-11*	32.54	3-11†	37.72
cop	2	28.86	7-5	77.29	4-5*	63.74	4-5†	67.18
obl	-3	18.77	1-2	23.79	1-3*	26.34	1-3†	25.97
case	1	43.49	7-5	70.27	3-9*	66.66	3-9†	67.27
mark	1	49.62	7-5	64.67	3-9*	63.41	3-9†	62.69
appos	-2	20.76	1-2	15.7	1-8*	18.89	6-1	21.56
punct	1	27.87	2-12	29.5	1-6*	23.92	6-7†	24.25
conj	-2	24.19	1-2	12.42	5-1*	29.47	5-1†	33.51
xcomp	-2	44.96	6-3	56.58	1-3*	64.3	1-3†	64.92

Figure 5.2: Top performing heads for dependency syntax recognition.

While no single head does particularly well at identifying dependencies overall (as indicated by the UNLABELED probe), there are heads that do very well at specific labeled dependency relations, sometimes out performing the baselines by a sizable margin. For example, there are heads in all three models that significantly out perform the baselines in identifying the "det," "obj," and "cop" relationships. Again, this is interesting that specific attention heads specialize for syntactical relationships, even though their pre-training task was unsupervised.

In terms of comparing the models, similar to part-of-speech tagging, for the most part the models have heads that perform similarly well on the tasks. There are some exceptions, however, such as differences in performance in the "flat," "compound," "nsubj," and "conj" relations where XLNet performs better than BERT, and in the "cop" relationship where BERT performs better than XLNet. Unlikely in part-of-speech tagging, the best performing heads of XLNet and GROVER are tend to be in each other's top-five, and often are the same one. This may suggest that this heads truly specialize in these dependency relations as their primary functions, where as heads that do best in part-of-speech tagging may not do it as the singular function and thus are liable to focus more on their other functions as the model fine-tunes.

Section 5.4

Unlabeled Dependency Parser

It is clear from §5.2 and §5.3 that individual attention heads specialize for specific syntactical relationships. It then becomes natural to ask how much syntactical information is possessed within the attention heads of the model as a whole. Way way to do this, as discussed in §5.1, is to combine the heads into a single classifier through a trained linear weighting. In this case, the supervised task is to probe for *unlabeled* dependency relations. That is, the goal of the probe is to identify each word of the input sequence's syntactical head, irrespective of what the dependency relationship is. The idea is that if this probing classifier does well, it would suggest that a substantial amount of syntactical knowledge is possess within the attention of the model.

The probe design used in this paper is the same one suggested by Clark et al. (2019). An input sequence is passed into the model, and the attention heads $\alpha^{\ell,h} \forall \ell \in [1, L], h \in [1, H]$ are extracted. There are two weight vectors $\mathbf{w}$ and $\mathbf{u}$ with one weight for each head in the model which are trained through supervised learned on the dependency-prediction task. The heads and weights are combined to produce a probability $p(i|j)$ for each pair of words i and j in the input sentence, where $p(i|j)$ is the probability of word i being word j's syntactic head. Formally, this is expressed as

$$p(i|j) \propto \exp\left(\sum_{\ell=1}^{L}\sum_{h=1}^{H} \mathbf{w}_{(L*(\ell-1)+h)}\alpha_{i,j}^{\ell,h} + \mathbf{u}_{(L*(\ell-1)+h)}\alpha_{j,i}^{\ell,h}\right) \tag{5.1}$$

5.4.1. Results

To establish a baseline, a right-branching method is used, where each word's syntactic head is predicted to be the word following it.

Model	**UAS**
Right-Branching	26.78
BERT Attention	44.12
XLNet Attention	46.07
GROVER Attention	45.80
GROVER Attention with XLNet Weights	44.43

Figure 5.3: Performance of simple dependency parsers based on the frozen attention heads of the models.

As shown in Figure 5.3, the simple probing classifiers substantially out perform the baseline, suggesting that each of the models' attention maps have a fairly solid representation of syntax encoded within them. All three models achieve similar performances, suggesting that no mode has a significantly better understanding of language. As an additional analysis, when the weights trained on XLNet's attention heads were used with GROVER's attention heads, there was not a significant decrease in performance, suggesting that overall the same heads in XLNet and GROVER attend to dependency relations in similar manners and that XLNet's encoding of linguistic knowledge was not significantly changed over the fine-tuning process.

Further, it is likely that these probes can achieve even better UAS scores than those that are displayed. Recalling from §5.1.1, the dependencies were generated automatically by a model which itself has a UAS of 87.06%. When Clark et al. (2019) built the same probe, the did so on top of the Penn Treebank (Marcus et al., 1993) annotated with Stanford Dependencies. When they did, the probe based on BERT's attention heads achieved a UAS of 61%. This paper did not have access to that dataset, so by training the probe on a less noisy training dataset, there is strong reason to believe that these probes would perform even better, and in particular the probes based on XLNet and GROVER would improve in a similar fashion to that based on BERT, given their similar levels of accuracy on the same dataset.

Chapter 6

Attention Head Similarity

Section 6.1

Clustering with Jensen–Shannon Divergence

It can also be useful to understand how similar the attention patterns of different attention heads are. For example, one may wish to know whether attention heads within the same layer act similarly or not. Thus, there is a need for a measure of similarity between attention heads. Just as with the measure of entropy is §3.4, since attention is distributed like the probability mass function of a discrete random variable, we can use a concept from discrete probability. In this case, it is the Jensen-Shannon divergence metric, which is a measure of the similarity between two probability distributions proposed by Lin in 1991, and it is based on the entropy equation (3.4). The Jensen-Shannon divergence between two discrete random variables X and Y, $JSD(X||Y)$, with the entropy of a single discrete random variable Y being $H(Y)$, is computed as

$$JSD(X||Y) = H\left(\frac{X+Y}{2}\right) - \frac{1}{2}\left(H(X) + H(Y)\right) \tag{6.1}$$

That is, the Jensen-Shannon divergence is equal to the entropy of the combination of X and Y less the average entropies of X and Y individually. Intuitively what this equation is saying is that if combining X and Y gives a new discrete random variable with more entropy than by averaging X and Y, then the distributions of X and Y are dissimilar from each other. Conversely, if X and Y have the same distributions, then combining them will give the same entropy as each variable on it's own, and the Jensen-Shannon divergence will be zero. In other words, the Jensen-Shannon divergence is a measure of how much new entropy is generated by combining two discrete random variables, and thus is a good measure of how dissimilar two probability distributions are.

With this measure in mind, we can measure the relative similarity between between two attention heads $\alpha^{\ell,h}$ and $\alpha^{m,g}$ by averaging the Jensen-Shannon divergence across all their rows (which are attention distributions for a single token) across all sentences in the corpus. Alternative measures of similarity include a cosine similarity between flattened arrays of the attention weights (Kovaleva et al., 2019). Formally, the relative distance between two attention heads $\alpha^{\ell,h}$ and $\alpha^{m,g}$, $dist(\alpha^{\ell,h}, \alpha^{m,g})$, can be computed as

$$dist(\alpha^{\ell,h}, \alpha^{m,g}) = \sum_{\mathbf{x}\in\mathbb{C}} \frac{1}{|\mathbf{x}|} \sum_{i=1}^{|\mathbf{x}|} JSD(\alpha^{\ell,h}(\mathbf{x})_i || \alpha^{m,g}(\mathbf{x})_i) \tag{6.2}$$

6.1.1. Results

First, the relative similarities (i.e. distances) are calculated between each pair of attention heads in a given model using Jensen-Shannon divergence. Then a multidimensional scaling (Kruskal, 1964) is used to give each head a two dimensional embedding such that the Euclidean distance between each pair of coordinates reflects (as closely as possible) the distance expressed in equation (7.2) between the heads that correspond to the coordinates.

The clustering metric used is the average Jensen-Shannon divergence between all heads in the same layer. The "clustering index" present on the graphs is the average layer-wise clustering metric across all layers in the model.

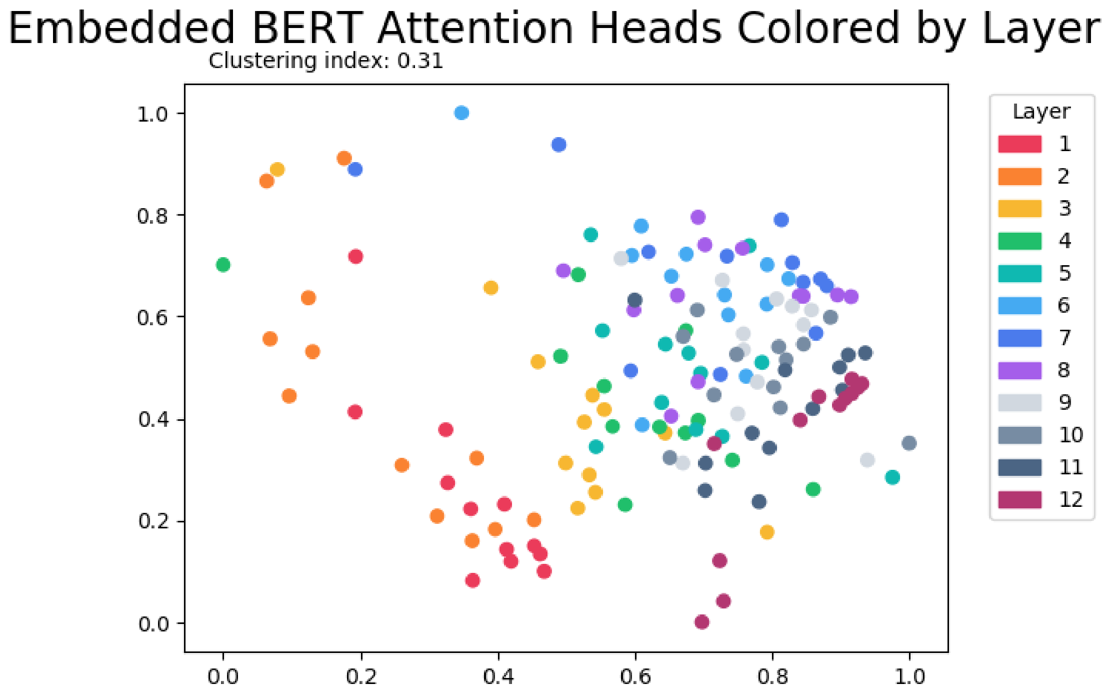

Figure 6.1: 2-D embedding of BERT's attention heads.

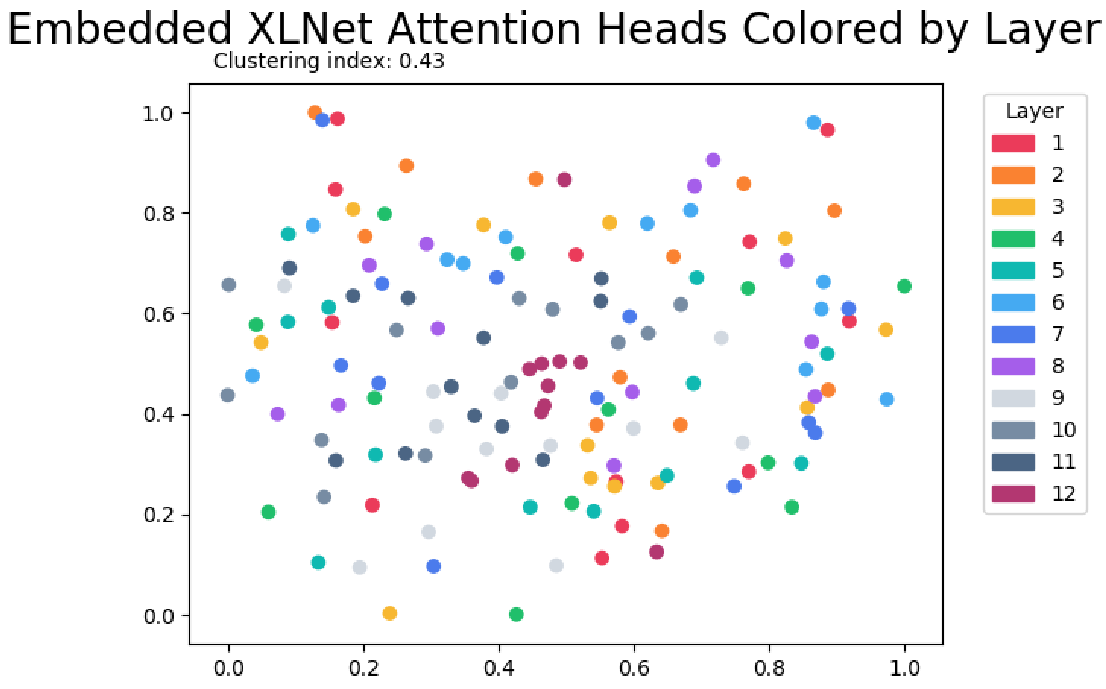

Figure 6.2: 2-D embedding of XLNet's attention heads.

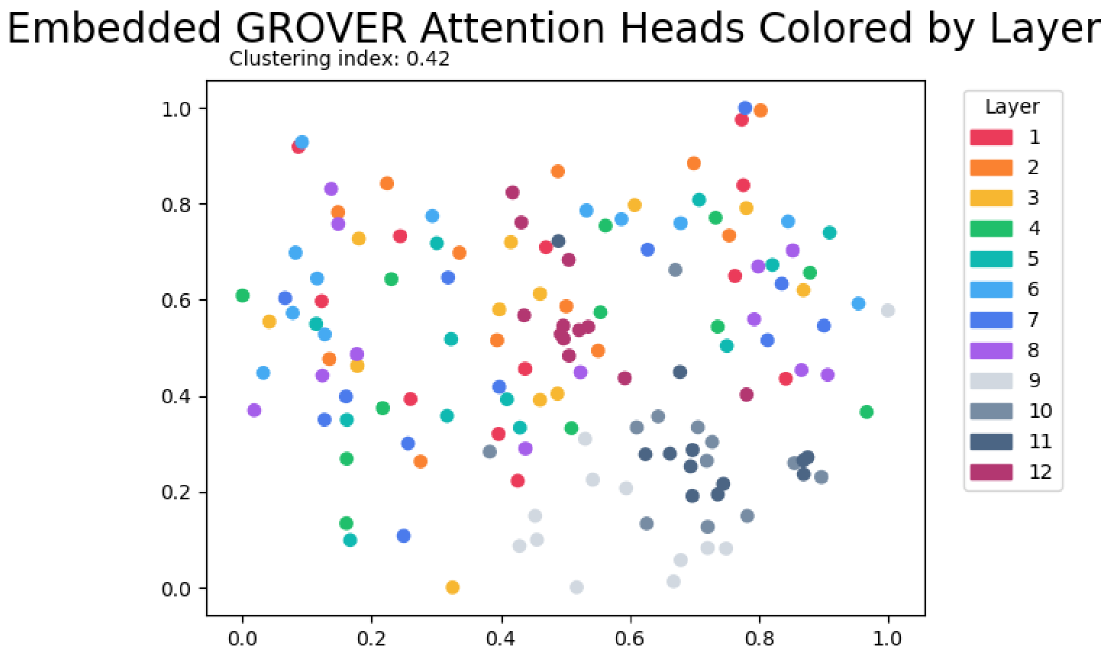

Figure 6.3: 2-D embedding of GROVER's attention heads.

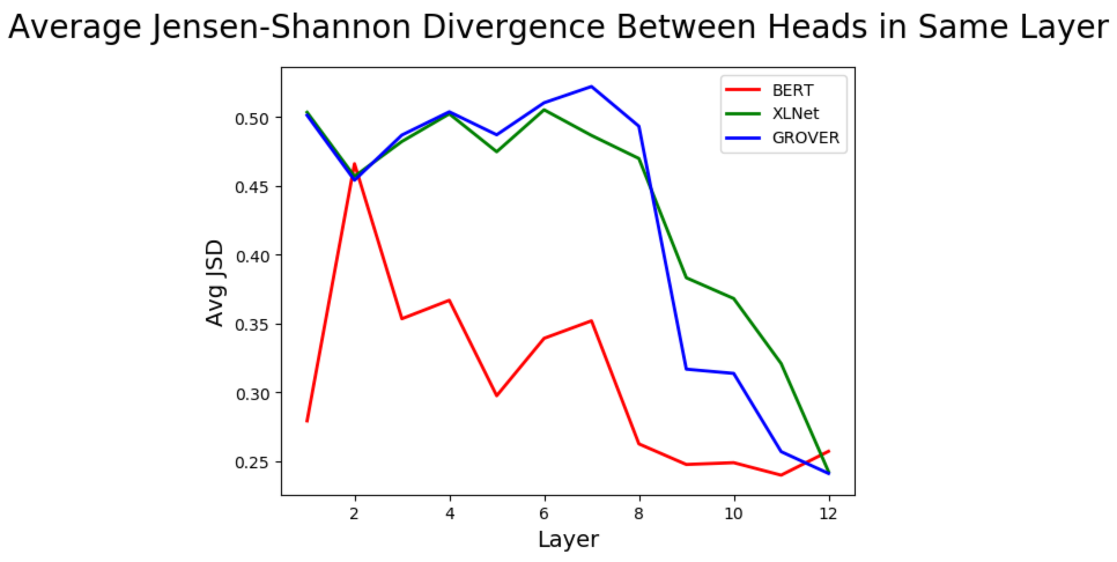

Figure 6.4: Average JSD between heads in the same layer across all layers of each model.

It can be seen qualitatively from Figures 6.1, 6.2, and 6.3 and quantitatively from Figure 6.4 that there does seem to be some clustering of attention heads by layer. This clustering is most prominent in the final layer of each model. Further, BERT's layers are more clustered than those of XLNet and GROVER. This means that compared to XLNet and GROVER, attention heads within the same layer of BERT tend to behave more similarly to each other, suggesting a higher level of redundancy in BERT's attention heads than in those of XLNet and GROVER. Clark et al. (2019) suggest that the reason for the apparent redundancy is the use of attention dropout during training, which causes some attention weights to be zeroed-out during training. However, this seems like a less plausible explanation now because XLNet was also trained with attention dropout but does not exhibit the same level of clustering.

Section 6.2 Changes Across Fine-Tuning

Rather than measuring how similar heads within the same model are, it is also interesting to see how similar heads are across two models. While it wouldn't make sense to compare head-for-head the attention heads between two different models such as BERT and XLNet, it does make sense to compare head-for-head between two versions of the *same* model, namely the base model compared to a fine-tuned version of the model. Seeing which heads change the most (relatively) across fine-tuning could give potential insights as to how the model adapts itself for a down-stream task.

6.2.1. Results

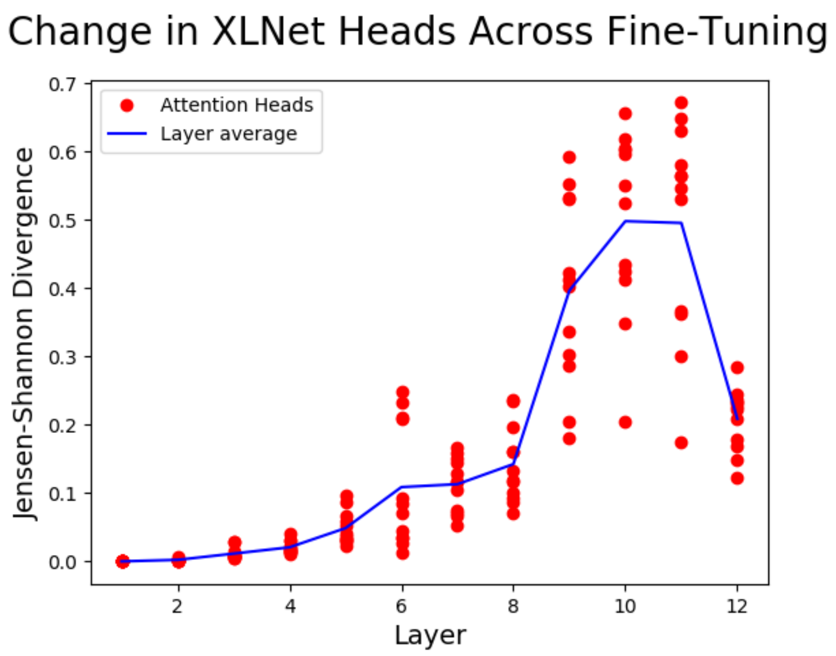

Figure 6.5: The Jensen-Shannon divergence between corresponding heads of XLNet and GROVER.

Figure 6.5 clearly shows how the vast majority of change over fine-tuning happens in the later layers of the model, particularly in layers 9, 10, and 11. There is almost no change at all in layers 1 and 2.

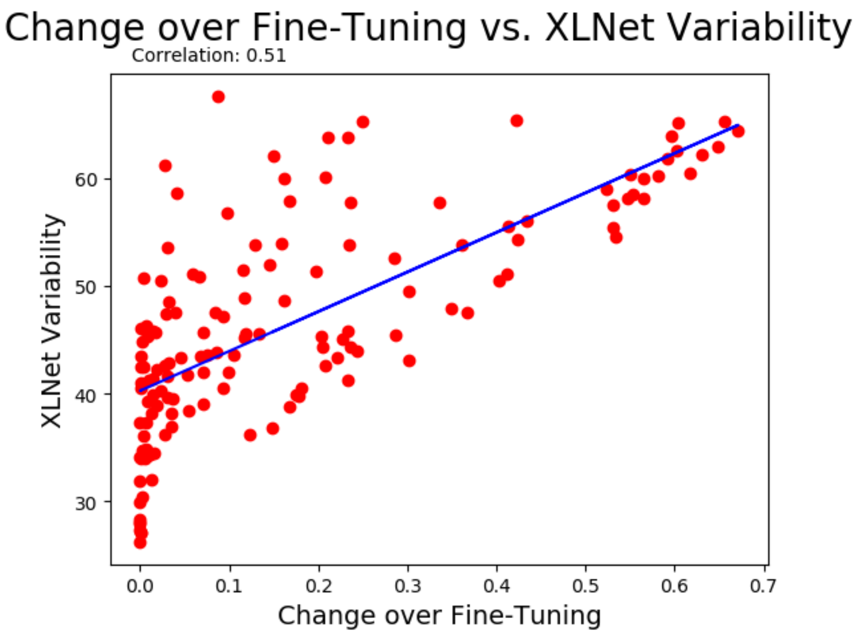

Figure 6.6: Attention head change over fine-tuning (measured by the JSD between a head in XLNet and GROVER) plotted against variability.

While the pattern of change is mostly described by which layer a head is in, there are still exceptions. For example, there are heads in layer 6 that changed more than some heads in layers 9, 10, 11, and 12. It then becomes natural to ask if there are any qualities of attention heads that makes them likely to change a lot over fine-tuning. After plotting the surface-level patterns discussed in §3, the amount an attention head changes over fine-tuning (as measured by the Jensen-Shannon divergence between itself in XLNet and GROVER) is most strongly correlated with its variability. As seen in Figure 6.6, while not every highly variable attention head changes a lot over fine-tuning, the attention heads that changed the most tend to be on the more variable side.

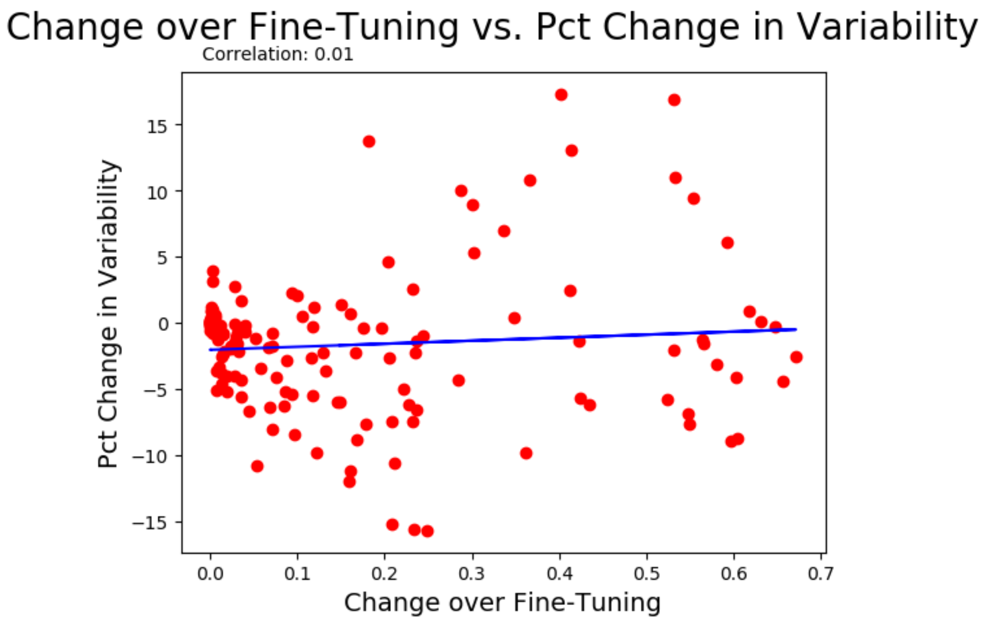

Figure 6.7: Attention head change over fine-tuning plotted against percent change in variability over fine-tuning.

Since it seems that variability correlates with how much an attention head changes over fine-tuning, it then becomes natural to ask whether those variable heads are changing to become more variable, which would suggest that perhaps they are becoming even more content-dependent for certain semantic knowledge relevant to the downstream task. However, Figure 6.7 shows that this is not the case. Of the attention heads that changed the most over fine-tuning, some increased in variability while some decreased in variability, with no clear pattern of change.

Chapter 7

Attention Head Importance

Section 7.1 Ablating Single Heads

It appears thus far that individual attention heads can specialize to different patterns, syntactical tasks, and semantic tasks. It is thus a natural question to ask how important any single head is to the model. The answer to this question can be explored through an ablation experiment.

An ablation experiment is one in which one or more attention heads from a given model are removed during a supervised test task. By evaluating the model's performance on the same task with different attention heads removed and comparing the differences, one can get a sense for how important an attention head (or combination of heads) is for that task. In order to "remove" an attention head from a model, one sets the outputs of that head to be all zeros. To formalized this, we must revisit equation (2.5) which was used in the definition of multi-headed attention. The only change is to introduce the variable $\xi_h \in \{0, 1\}$ and to rewrite (2.5) as

$$Z_h = \xi_h \text{Attention}(X, W_h^Q, W_h^K, W_h^V) \tag{7.1}$$

ξ_h is called the head's mask, and when it is 0 it is equivalent to setting the output of head h within the multi-head attention mechanism to all zeros (thus the same as removing it). When it is 1, this equation is equivalent to (2.5) and the attention head functions normally.

7.1.1. Results

Layer \ Head	1	2	3	4	5	6	7	8	9	10	11	12
1	-1.67	-4.17	-0.83	-1.67	-1.67	-2.5	-4.17	-2.5	-2.5	-0.83	-2.5	0.0
2	-3.33	-1.67	-0.83	-3.33	-2.5	-0.83	-2.5	-0.83	-2.5	-2.5	-4.17	-2.5
3	-3.33	-2.5	0.0	-0.83	-0.83	-1.67	-0.83	-1.67	-0.83	-2.5	-1.67	-0.83
4	-4.17	-2.5	-3.33	-0.83	0.0	-0.83	-1.67	-5.0	-5.0	-3.33	-2.5	-2.5
5	-0.83	-0.83	-0.83	-2.5	-3.33	-0.83	-2.5	-1.67	-0.83	1.67	-3.33	2.5
6	-1.67	-1.67	-0.83	0.0	-2.5	-0.83	-1.67	-5.83	-3.33	-0.83	-0.83	0.0
7	-4.17	-1.67	-1.67	-4.17	-2.5	-3.33	-5.83	-0.83	-1.67	-1.67	-4.17	-2.5
8	-5.0	-3.33	-5.0	-1.67	-1.67	-3.33	-4.17	-3.33	-3.33	-2.5	-3.33	-1.67
9	-3.33	-2.5	-3.33	-0.83	-0.83	-0.83	-5.0	-4.17	0.0	-1.67	-5.0	-0.83
10	**-12.5**	-2.5	-3.33	-1.67	-3.33	-2.5	0.0	-2.5	0.0	-0.83	-0.83	-1.67
11	-2.5	0.0	-4.17	-3.33	-4.17	-2.5	-2.5	-2.5	-3.33	-3.33	-3.33	-3.33
12	**-6.67**	-2.5	-1.67	-3.33	-2.5	-3.33	-2.5	-2.5	-2.5	-3.33	-0.83	-1.67

Figure 7.1:

Overall, ablating any given head does not affect GROVER's performance drastically, with the vast majority of heads affecting accuracy by 5 percentage points or less (with a few exceptions). In some cases, such as heads 5-10 and 5-12, ablating heads can even *increase* the model's performance. Further, there are many heads whose ablation does not affect the model's performance at all. This suggests that there may be redundancies built into the model's attention heads.

The two heads that cause the largest loss of accuracy when ablated are heads 10-1 and 12-1. 10-1 is the second best head at identifying the determiner part-of-speech (with an accuracy of 41.84). However, the best head at that relationship, 5-6 (with an accuracy of 42.63) when ablated only affects GROVER's performance by -0.83 percentage points. But in addition to doing well at identifying determiners, head 10-1 also does well at identifying nouns (with an accuracy of 53.6, compared to the best head's accuracy of 55.41) and adpositions (with an accuracy of 47.61, compared to the best head's accuracy of 48.97). Head 10-1 is not in the top 5 of GROVER's heads for paying attention to any hate-related words, so perhaps ablating it affects the model mostly through loss of syntactical knowledge rather than semantic knowledge. Head 12-1, on the other hand, is the attention head that pays the most attention to the "n-word", suggesting that ablating it may affect GROVER's classification accuracy by loss of semantic knowledge about the hate-speech-detection task in the final layer of the model.

Section 7.2
Ablating Multiple Heads

After ablating single heads and seeing that for many of them their removal from the model does not affect GROVER's performance by much, it then begets the question *how many* heads can be removed without drastically impacting performance. In order to test the compounding effect of pruning multiple heads, this paper uses the iterative ablation approach suggested by Michel et al. (2019). Since trying every combination of head masks is not practical given the number of heads in the model and how long each evaluation takes, a heuristic approach is taken to iteratively select the next head to ablate.

The heuristic used is a proxy for a head h's importance, which is estimated as the model's expected sensitivity to that head's mask variable ξ_h. With the loss function $\mathcal{L}$, the model's expected sensitivity to a head h, I_h, can be estimated as

$$I_h = \sum_{\mathbf{x}\in\mathbb{C}} \frac{1}{|\mathbf{x}|} \left| \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \xi_h} \right| \tag{7.2}$$

Michel et al. (2019) found it was important to take the absolute value of the head mask's gradient to keep from highly negative and positive contributions from canceling each other out in the sum. Further, the importance scores are normalized by layer using the ℓ_2 norm as suggested by Molchanov et al. (2017). Comparing the importance scores across all non-ablated heads gives a heuristic ranking of the heads in order of importance. The least important head is the next one to be ablated. Further, the order in which the heads are ablated can be used as an overall ranking of head importance in the model.

7.2.1. Results

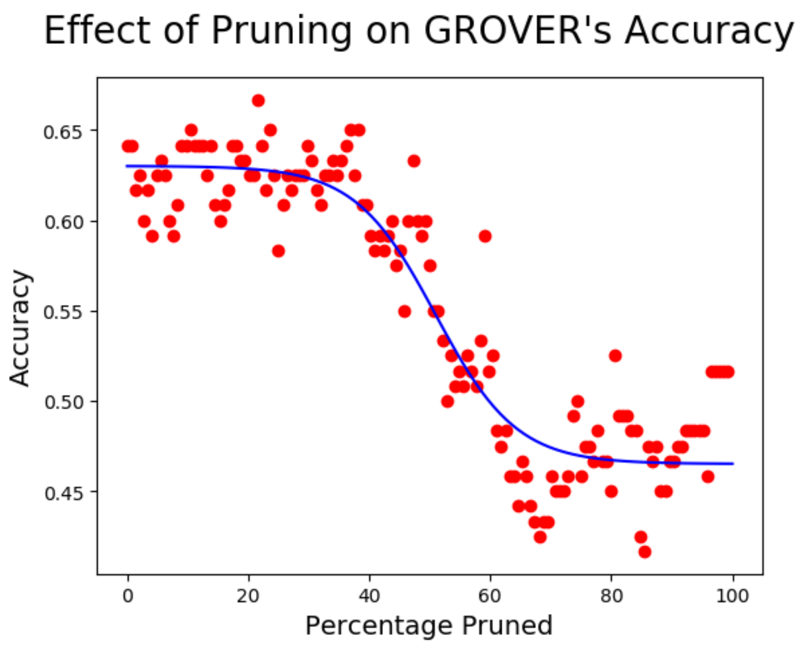

Figure 7.2: GROVER's performance as its heads are pruned according to the heuristic expressed in equation (7.2).

Interestingly, almost 40% of GROVER's attention heads can be pruned without a significant negative impact on its performance accuracy. This suggests that there is some redundancy built into the model. Further, performance drops sharply past a certain point, suggesting that while many heads can be pruned, the majority of attention heads are still important.

Section 7.3

Changes in Head Importance Across Fine-Tuning

Armed with measure of relative head importance from §7.2's experiment, this paper attempts to see if head importance correlates with any other attention-head qualities discussed thus far.

7.3.1. Results

R^2	Change over Fine-Tuning	XLNet Entropy	XLNet Variability	Change in Entropy	Change in Variability	Head Importance
Change over Fine-Tuning	1	.045	.514	.086	.007	.062
XLNet Entropy	.045	1	.099	.040	.027	.027
XLNet Variability	.514	.099	1	.053	.020	.004
% Change in Entropy	.086	.040	.053	1	.178	.005
% Change in Variability	.007	.027	.020	.178	1	.001
Head Importance	.062	.027	.004	.005	.001	1

Figure 7.3: R^2 correlation scores between different metrics for attention heads.

R^2	Head Importance
GROVER Entropy	.024
GROVER Variability	.006
Change in GROVER Accuracy from Ablation	.084

Figure 7.4: Some additional R^2 correlation scores

Head importance, as measured by this paper, does not correlate strongly with any of the listed variables. The only two variables in Figure 7.3 that correlate somewhat are variability of XLNet heads and how much they change over fine-tuning, and this was discussed in §6.2.1. This would suggest that there may not be a single macro-quality of an attention head that determines its importance. It may be that importance is a result of more specific qualities, such as attention to dependency relations or task-specific knowledge such as head 12-1's attention to the "n-word." In fact, head 12-1 is the fifth most important head according to the heuristic used by this model. Another example is manual inspection of head 7-7, which was the second most important head according to the importance heuristic, places in the top five heads for attending to both the words "whites" and "gay." However, the most important head, 9-9, does not seem to specialize in any semantic or syntactic pattern quantified by this paper.

Chapter 8

Discussion

Section 8.1

Summary of Findings

8.1.1. BERT vs XLNet

The vast majority of functional and linguistic phenomena studied in this paper appear to be present in both BERT and XLNet to similar extents. Many of the surface level patterns studied in §3 are very similar between the models, with layer averages behaving similarly, with similar frequencies and extents of individual heads specializing for specific patterns. This is true for patterns of relative position (§3.2), entropy (§3.4), and variability (§3.5). The largest difference was in how the models take in bidirectional context, with XLNet attending in both directions relatively evenly throughout all layers, while BERT pays most of its backwards attention in the first three layers and then attends mostly forward in the other layers (§3.3). The models also react similarly to specific input tokens, with XLNet responding to periods in the way that BERT responds to `[SEP]` tokens (§4.1, §4.2).

Further, linguistic knowledge seems to be present in both models to similar extents. In general, both models seem to at least partially organize their internal structures in accordance with the linguistic features which we intuitively believe are important, which is interesting given that they were never supervised on such features. The probing for knowledge of syntactic features such as parts-of-speech (§5.2) and dependency relations (§5.3, §5.4) showed that linguistic knowledge across the board was present in similar levels in the attention heads of the models, and certainly no model had a clear advantage.

So then it begets the question: if the two models have similar architectures, the same training hyperparameters, were trained on the same data, their attention heads function similarly, and neither model seems to have a significantly richer representation of language, what accounts for the performance difference? It is likely a number of factors. The first may be that XLNet's pre-training task eliminates the pre-train/fine-tune discrepancy present in BERT because of the use of `[MASK]` tokens to corrupt the training inputs. The second may be XLNet's auto-regressive formulation. A third may be how XLNet does a better job of attending bidirectionally throughout its layers.

An additional fourth explanation that seems plausible based on the findings of this paper is because of XLNet's lower level of redundancy compared to BERT. Figure 6.4 showed how BERT's attention heads tend to be more similar to other attention heads in their layer than the attention heads of XLNet. This suggests a higher level of redundancy present in BERT. However, Li et al. (2018) showed that encouraging diversity among attention heads in the Transformer model through disagreement regularization can improve performance on translation tasks. It may be that a side-effect of XLNet's AR architecture and different pre-training task causes its attention heads to be more diversified, and this diversification in turn may contribute to a greater ability to fine-tune on downstream tasks.

8.1.2. XLNet vs GROVER

While there are clear differences between XLNet and GROVER, the differences exist in a predictable manner. The vast majority of difference between the attentions of XLNet and GROVER exist in the deeper layers, as illustrated by Figure 6.5. In particular, the greatest changes happen in layers 9 and 10. The interesting thing is that the differences often seen between XLNet and GROVER that present themselves in these layers have to do with *semantic* information. As a few examples, Figure 3.8 shows that layer 9 had the largest increase in variability, suggesting that it may be becoming more content-dependent. Additionally, Figures 4.11 and 4.12 show that it was later layers that really increased in their attentions to task-specific words such as "jew" and "gays." Further, Figure 4.3 shows an increase in attention to <`cls`> tokens in layers 8, 9, 10, and 11; whereas XLNet pays uniform attention to these tokens, GROVER significantly increases its attention to them, in particular in layer 9, suggesting that these are the layers primarily attending to the classification task of GROVER.

Throughout all these changes, there did not seem to be much change in how the model encodes its linguistic knowledge. §5.2 and §5.3 showed that similar levels of parts-of-speech recognition and dependency relationship knowledge were present in both models, with §5.3 in particular showing that it was largely the same heads in both XLNet and GROVER attending to specific dependency relations. This idea is further bolstered in §5.4 showing how the trained weights for the dependency parsing probe using XLNet's attention heads transfer very well to GROVER's attention heads, meaning attention heads important for dependency parsing in XLNet are similarly important in GROVER.

In all, it seems that GROVER changed in semantic knowledge compared to XLNet, but not in syntactic knowledge. It's clear that much of the semantic knowledge of GROVER is possessed by the deeper layers, and it can similarly be deduced that much of the syntactic knowledge of XLNet and GROVER is possessed by the earlier layers, which did not change much over fine-tuning. This idea is supported in the literature, such as Tenney et al. (2019) indeed finding that the layers of Transformer-based models perform the classic NLP pipeline in order, starting with part-of-speech tagging, then dependency parsing, then name-entity recognition, and then semantic roles in the later layers.

Section 8.2

Limitations

There were two main limitations to this work, both having to do with annotated datasets used. The first is that the dataset GROVER was trained on was not of the highest quality. Manual inspection of the data showed that there were certain authors who were clearly hate-speech-spreaders not labeled as such. This made it difficult to train GROVER to a high validation accuracy, since the data itself was fairly noisy. This may have kept GROVER from learning certain bits of semantic knowledge which we intuitively know to be important to such a task. As such, some meaningful representations that should have been present in GROVER may not have been there. Overall, GROVER is a fragile model that is extremely sensitive to the inputs it is given, so while the changes in fine-tuning studied in this paper through GROVER are likely valid, it would be good to see if such patterns are present when XLNet is fine-tuned for other tasks with better training data.

The second main limitation as mentioned was that the corpus was annotated with linguistic features automatically by another model. This model does fairly well, particularly for part-of-speech tagging, but using an actually-annotated dataset would give more credibility to the findings.

Section 8.3

Future Work

The best way to bolster the findings of this paper would be to introduce a number of other fine-tuned models. A caveat to the analyses in this paper is that only XLNet was fine-tuned, and only on one downstream task. In order to increase confidence that certain phenomena occur during fine-tuning, multiple fine-tuning experiments should be done and compared, such as to truly discern whether higher variability lends itself to larger changes during fine-tuning.

Additionally, it would be interesting to fine-tune both BERT and XLNet on the same tasks to see if there are any differences in how they fine-tune themselves. The analyses in this paper are well-defined and seem to give interesting, interpretable results, but future work is need to apply these same analyses across a large set of fine-tuning tasks in order to be confident that the patterns observed are not specific to any single downstream task but to the model mechanics themselves.

In particular, an interesting question to ask would be whether input length plays a role in how a model fine-tunes. Recalling from §2.4.2, when fine-tuning GROVER it was never given an input sequence longer than 512 tokens, thus never triggering XLNet's auto-regressive mechanism. Further, there are a number of instances where GROVER seems to abruptly depart from XLNet's behavior *towards* that of BERT. Notable examples include the shift towards attending more to the forward context in later layers like BERT as illustrated by Figure 3.4 and increasing its attention to separator tokens like BERT and decreasing its attention to periods as illustrated by Figures 4.1 and 4.2. It would be interesting to see if this is coincidence, or if part of the differences between BERT and XLNet can be explained by the *length* of the inputs they are trained on or the AE vs AR formulations of the models, and if fine-tuning on shorter-length inputs that do not trigger the AR mechanism causes fine-tuned versions of XLNet to become more like BERT.

A final interesting area to study would be how models fine-tune not just to incorporate task-specific semantic knowledge, but to task-specific domains. In this case "domain" refers to differences in overall language trends. GROVER, for example, works on social media text, which is written in a much different style than snippets from Wikipedia. It would interesting to measure what proportion of a model's change over fine-tuning is a from learning task-specific semantic knowledge, and what proportion is from the model attempting to map its linguistic knowledge onto a slightly different domain.

Chapter 9

Conclusion

As machine learning models grow in complexity and capability, they will be applied in increasingly significant ways. However, as their ability to make an impact increases, so too do the consequences of their decisions increase in severity. Machine learning is no longer just deciding what movie to recommend to you next on a streaming platform, but now may be driving the car next to you on the road. Thus, as machine learning increases in capability, so too does the capability to interpret models need to increase. With this philosophy in mind, as machine learning models are designed for real-world applications, their interpretability will have to be considered just as much as their performance.

Fortunately, it seems that the highest-performing models for NLP tasks are also naturally interpretable thanks to the attention-mechanisms they are built upon. As shown throughout this paper, attention can be used to understand a tremendous amount about the inner workings of an attention based model, spanning from how models organize their internal structures to the specific functional and semantic knowledge they possess. For this reason, model designers should focus on pushing attention-based models to their limits in performance and scope.

Bibliography

[1] Alammar, J. The illustrated transformer. http://jalammar.github.io/illustrated-transformer/.

[2] Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., and Zhang, L. Bottom-up and top-down attention for image captioning and visual question answering, 2018.

[3] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate, 2016.

[4] Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. Attention-based models for speech recognition, 2015.

[5] Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. What does bert look at? an analysis of bert's attention, 2019.

[6] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.

[7] Devlin, J., Chang, M., Lee, K., and Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR abs/1810.04805* (2018).

[8] Foundation, W. Natural language processing. https://en.wikipedia.org/wiki/Natural$_{l}anguage_{p}rocessing$.

[9] Foundation, W. Wikimedia downloads. https://dumps.wikimedia.org.

[10] Goldberg, Y. Assessing bert's syntactic abilities, 2019.

[11] Hewitt, J., and Manning, C. D. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 4129–4138.

[12] Kovaleva, O., Romanov, A., Rogers, A., and Rumshisky, A. Revealing the dark secrets of bert, 2019.

[13] Kruskal, J. B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika 29*, 1 (1964), 1–27.

[14] Larson, J., Mattu, S., Kirchner, L., and Angwin, J. How we analyzed the compas recidivism algorithm. *ProPublica* (May 2016).

[15] Li, J., Tu, Z., Yang, B., Lyu, M. R., and Zhang, T. Multi-head attention with disagreement regularization, 2018.

[16] Lin, J. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory 37*, 1 (1991), 145–151.

[17] Lotter, W., Diab, A., Haslam, B., and et al. Robust breast cancer detection in mammography and digital breast tomosynthesis using an annotation-efficient deep learning approach. *Nat Med 27* (2021), 244–249.

[18] Luo, F., Yang, P., Li, S., Ren, X., and Sun, X. Capt: Contrastive pre-training for learning denoised sequence representations, 2020.

[19] Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics 19*, 2 (1993), 313–330.

[20] Michel, P., Levy, O., and Neubig, G. Are sixteen heads really better than one?, 2019.

[21] Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. Pruning convolutional neural networks for resource efficient inference, 2017.

[22] Molnar, C. *Interpretable Machine Learning.* 2019. https://christophm.github.io/interpretable-ml-book/.

[23] Pennington, J., Socher, R., and Manning, C. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Doha, Qatar, Oct. 2014), Association for Computational Linguistics, pp. 1532–1543.

[24] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations, 2018.

[25] Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (2020).

[26] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. Tech. rep., OpenAI, 2018.

[27] Raganato, A., and Tiedemann, J. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop*

BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP (Brussels, Belgium, Nov. 2018), Association for Computational Linguistics, pp. 287–297.

[28] Rogers, A., Kovaleva, O., and Rumshisky, A. A primer in bertology: What we know about how bert works, 2020.

[29] Shannon, C. E. A mathematical theory of communication, 1948.

[30] Taylor, W. L. Cloze procedure: A new tool for measuring readability. *Journalism Bulletin 30*, 4 (1953), 415–433.

[31] Tenney, I., Das, D., and Pavlick, E. Bert rediscovers the classical nlp pipeline, 2019.

[32] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2017.

[33] Vig, J., and Belinkov, Y. Analyzing the structure of attention in a transformer language model, 2019.

[34] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding, 2020.

[35] Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, 2015.