Augmenting GPT-2 with Hierarchical Predictive Processing for Story Ending Prediction

Erik S. McGuire
erik.s.mcguire@gmail.com

Abstract

Hierarchical predictive processing (or predictive coding; PC) is a unifying neurocognitive paradigm and interdisciplinary research program which has been taken up in numerous domains, including computational modeling and subdomains of AI, such as Natural Language Processing (NLP). One type of NLP task in recent years is predicting the endings to brief stories, called the Story Cloze Test (SCT); this is primarily intended to examine the modeling of causal relations. We ask whether we can augment GPT-2, a large pretrained causal language model, with predictive coding in order to improve performance on SCT, formulated as a multiple-choice task. To do so, we add a contrastive PC loss to the conventional language modeling (LM) and multiple choice (MC) losses of the SCT task. This auxiliary loss is based on top-down prediction errors between layers' latent representations of sentences in stories. Results averaged across numerous seeds suggest that PC can be applied to GPT-2 as a substitute for or as a complement to language modeling to improve SCT task performance.

1. Introduction

Hierarchical predictive coding (PC) is a neurocognitive paradigm which has become prominent in recent years, including the field of AI, and is notably used to explain phenomena (such as human cognitive biases and illusions) that have been difficult to reconcile with traditional views of neurocognitive processing. For computational modeling and artificial neural networks it has been suggested as more 'biologically plausible' (Whittington and Bogacz, 2017) than conventional formulations of backpropagation-based learning. In recent years, PC accounts of language processing have been researched in conjunction with deep learning-based approaches to NLP, demonstrating compatibility (Schrimpf et al., 2020).

A key downstream domain of tasks for NLP explores processing spans of text across utterances or sentences, such as discourse or narratives. Narratives–such as brief stories–allow for the investigation of how well causal relations between entities are modeled. Transformer models with self-attention such as BERT and GPT-2 have been suggested to learn hierarchical linguistic representations (Tenney et al., 2019); in this work we ask whether we can augment a large pre-trained generative language model, GPT-2 (Radford et al., 2018) with mechanisms inspired by predictive processing in order to learn sentence-level representations for predicting story endings. To do so, we follow similar methods to Araujo et al. (2021), who explored the impact of predictive processing on BERT-type models by adding top-down connections from upper to lower layers in order to compute a loss intended to be complementary to that derived from bottom-up predictions in the standard model.

We evaluate results on the Story Cloze Test (SCT; Mostafazadeh et al. 2016), a task that uses a story plot to predict the ending, in order to assess the downstream impact of this additional predictive loss. In this way, we hope to understand whether a hybrid of predictive processing and backpropagation can expand generative transformer models' representations.

2. Related Work

2.1. Predictive Coding

In backpropagation, during a forward pass, each layer receives the activations of the layer below it, and, during the backward pass, starting from the top, upper layers pass down the prediction error to earlier layers, fractionally assigning blame for a global loss to be minimized. This is analogous with the traditional view of the brain as passively receiving and processing sensory information before finally producing predictions and actions, and obtaining corrective feedback.

Predictive coding, however, minimizes local losses rather than a global loss, and flips the feedforward and backpropagation view so that the process begins with higher-level predictions: starting the 'forward' pass at the top, any given layer does not receive the activations of the layer below as input: instead, it predicts the input from

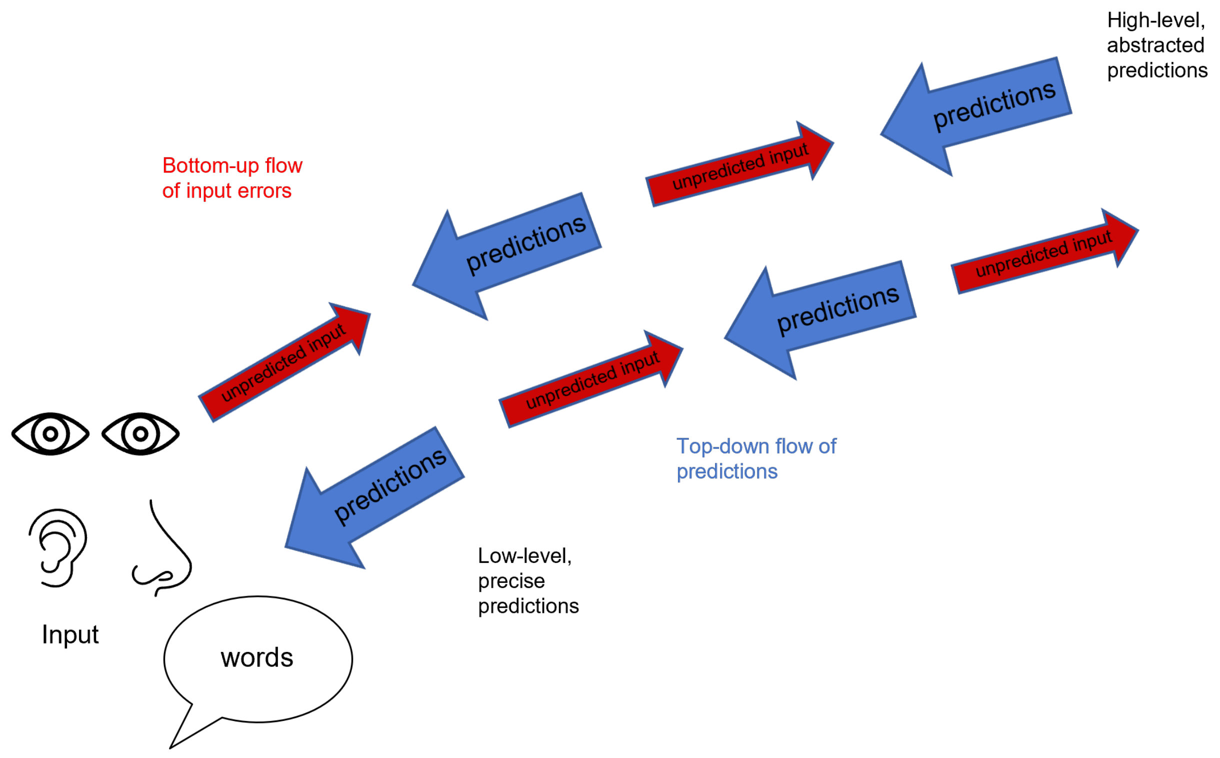
High-level, abstracted predictions
predictions
unpredicted input
Bottom-up flow of input errors
predictions
unpredicted input
predictions
unpredicted input
unpredicted input
Top-down flow of predictions
predictions
Low-level, precise predictions
Input
words

Figure 1: A schematic of the hierarchical information flow suggested by predictive processing accounts. Predictions of the inputs below are transferred top-down, and errors from these inputs proceed in bottom-up fashion, so that subsequent hypotheses from the upper layers are updated with this feedback; figure recreated from Lupyan and Clark (2015).

Story Body: 1. <s>John was writing lyrics for his new album. <sep> 2. He started experiencing writer's block. <sep> 3. He tried to force himself to write but it wouldn't do anything. <sep> 4. He took a walk, hung out with some friends, and looked at nature. <sep>
Gold Ending: 5a. He felt inspiration and then went back home to write. <sep>
Wrong Ending: 5b. John then got an idea for his painting. <sep>

Table 1: Example story body of four sentences, with the ground truth ending and an incorrect ending (in this case, containing an irrelevant reference to painting). The task is to choose the label/index (0 or 1) of the correct ending. Special tokens added during preprocessing are shown in brackets.

the layer below, and in turn receives its prediction error from that lower target layer.

In the context of deep learning, both pure and hybrid models of PC have been proposed in recent years (Millidge et al., 2021), where more pure models replace backpropagation altogether (Millidge et al., 2020), whereas PC-inspired models such as PredNet (Lotter et al., 2016) or Contrastive Predictive Coding (CPC; Oord et al. 2018) make use of various aspects of the different PC formulations to improve unsupervised learning of representations for a variety of spatial and temporal tasks, including NLP. We can differentiate between temporal, autoregressive predictive coding, and hierarchical predictive coding. These mechanisms operate in the latent space of networks' layers, rather than modeling in the data-space, described as less wasteful (Oord et al., 2018), and more congruent with the Rao-Ballard (Rao and Ballard, 1999) paradigm of predictive coding (Millidge et al., 2021).

Chung and Glass (2020) use autoregressive predictive coding or APC, predicting future observations in the latent space to improve speech representation learning with a transformer-based network. Hierarchical predictive coding (Hosseini and Maida, 2020) entails the higher layers in a hierarchy predicting inputs from lower layers, similar to the top-down connections of autoencoders

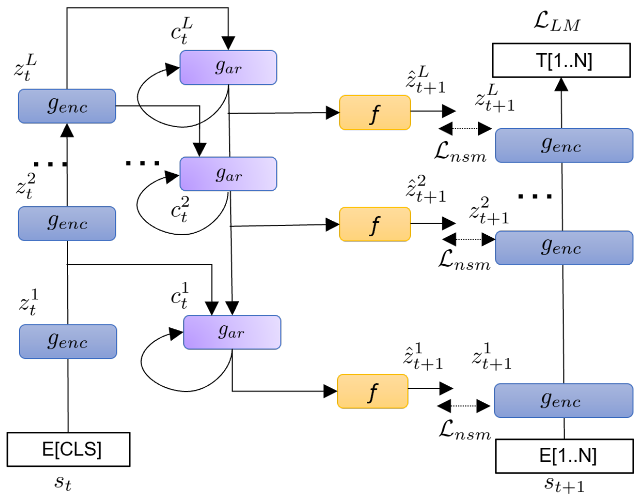

Figure 2: Top-down predictive loss computation diagram recreated from Araujo et al. (2021). Starting from the top down, from BERT hidden states g_{enc}, sentence representations z_t^ℓ are taken and sent through a GRU g_{ar} to obtain context c_t^ℓ; c_t^ℓ is transformed to prediction $\hat{z}_{t+1}^\ell$ (the subscript referring to the next timestep being predicted) for error computation against the previous layer's BERT encoding sentence representation z_{t+1}^ℓ of the next timestep.

or restricted Boltzmann machines (Hinton, 2007); this may be combined with temporal PC (the prediction of the next step in a sequence; Singer et al. 2019). While they do not predict previous layer representations, Araujo et al. (2021) have recently augmented BERT-based architectures to make use of features of PredNet and CPC, namely the top-down feedback connections of PredNet and the contrastive loss of CPC. In their work, where each input is a contiguous set of sentences, Araujo et al. take the BERT layers' encodings and select the [CLS] encodings as the layers' sentence representations, inserting an additional autoregressive recurrent network to further model the sequence at the sentence level so that discourse properties can be captured more effectively.

CPC (shown in Figure 3) does not operate across layers: the prediction and loss are calculated within the same layer. That is, the prediction of a layer at a timestep targets that same layer's representation of the next step. This intra-layer temporal prediction and loss is the method inherited by Araujo et al., as well: a sentence encoding in layer ℓ is used to predict the next sentence encoding in ℓ.

Araujo et al. (2021) as well as Lotter et al. (2016)'s PredNet combine the upper layer's context vector with the current layer's in order to make predictions; the use of the above layer's context vector as a top-down connection to the below layer's representation is a quirk of PredNet that deviates from the standard Rao-Ballard (Rao and Ballard,

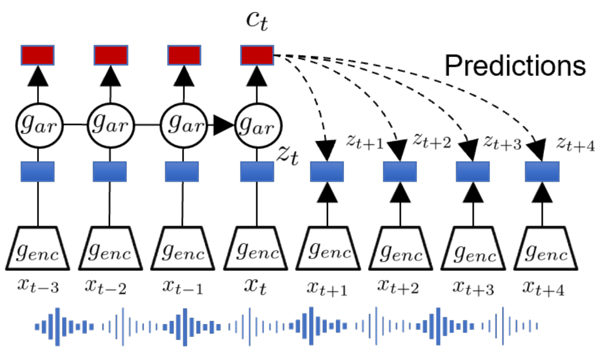

Figure 3: Contrastive predictive coding with $k = 4$ predictions ahead of the same layer's subsequent timesteps; figure recreated from Oord et al. (2018).

1999) notion of PC (Hosseini and Maida, 2020). In PredNet, which uses convolutional recurrent units, a layer's latent representation is derived from both its previous hidden state and the representation from the layer above (as seen by the arrows pointing down to and from the green *Representation* box in the diagram in Figure 4). Conventionally, as seen via the blue *Prediction* and *Target* boxes' error, the only top-down aspect is the prediction, where the layer predicts the target layer below.

Thus, the hybridity of predictive coding and deep learning and the variation in the PC implementation in Araujo et al. involves starting with feedforward encodings from BERT where each layer's encodings are built from the previous layer's: this violates the PC notion that a layer does not receive the representation below as input, it only receives the error information, which is associated with energy efficiency (Ali et al., 2021); it also differs in how top-down connections are used to something akin to PredNet's, and a change in predictions in loss from descending inter- to laterally intra- layer, as well as the backward propagation of errors, rather than PC's bottom-up sweep.

Our view is that the lack of straightforward reasoning or supporting research behind these variations in deep learning PC leaves room for exploration. To begin with, we choose to research GPT-2, rather than BERT, so that the more cognitively analogous (Schrimpf et al., 2020) autoregressive encodings can be used, reducing reliance on an additional recurrent model, and so that we can easily transition to exploring effects on Natural Language Generation in future work. Links between GPT-2 and human predictive language processing have been established by Heilbron et al. (2021), using GPT-2 to quantify the hierarchical word-level and phoneme-level predictions evidenced by the brain recordings of human subjects. We elect to use conventional top-down feedback in the form of predicting previous layers' inputs, either predicting the previous layers' representation of the same timestep and/or temporally (next timestep).

2.2. ROC Stories

The ROC Stories corpus and its Story Cloze Test (SCT) is a task introduced by Mostafazadeh et al. (2016) intended to assess and improve the learning of causal relations and other aspects of commonsense reasoning. This involves correctly predicting the endings of brief stories, typically by modeling not only subword and word-level tokens but sentence-level and story-level representations. We find the SCT task a natural fit for studying the impact of predictive coding mechanisms on GPT-2.

Radford et al. (2018) improved on the coeval state-of-the-art on the Story Cloze Task through multi-task fine-tuning of GPT-2. That is, the model was fine-tuned with the joint losses of the Story Cloze Task and an auxiliary language modeling task. The language modeling objective was used to improve generalization and accelerate convergence. The input was structured in this work by bracketing stories with a special start and end token and simply delimiting the story body (the first 4 sentences) from the ending (the final sentence).

However, Cai et al. (2017) have noted that annotation artifacts enable models to predict endings effectively with minimal input, using superficial features as clues. In this work, experiments include the modeling of individual sentences in stories, rather than delimiting only the body from the ending. Our early experiments on the SCT task found this to be a more effective structuring of the input for both baseline and PC models.

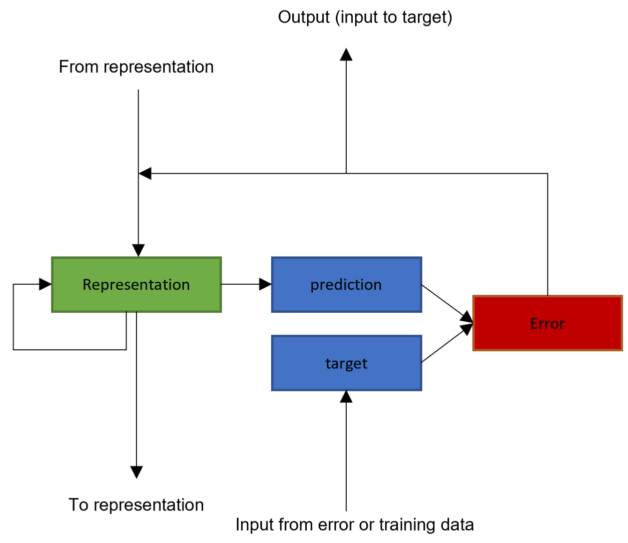

Figure 4: PredNet schematic of recursive, top-down, and bottom-up connections for a given layer representation; figure recreated from Lotter et al. (2016). Note the atypical use of direct connections between representations.

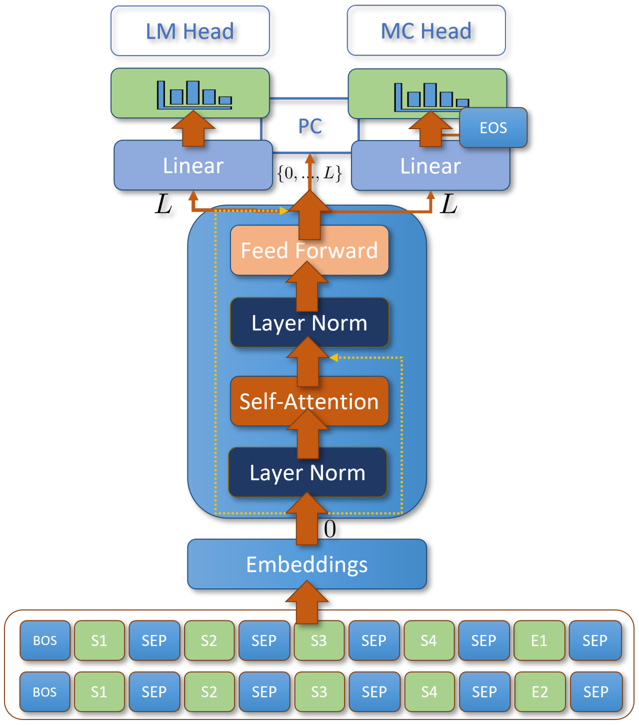

Figure 5: The standard preprocessing stage constructs a version of each story by concatenating the labeled story with the respective correct/incorrect endings, which are delimited from the story by a special token embedding. Final layer end-of-sequence (EOS; SEP in our case) token encodings are extracted as story representations for classification, and in the multiple choice head the vectors for the two choices are projected through a linear layer to obtain prediction scores to normalize with a softmax layer. LM preprocessing uses the same dual versions of the stories, setting the labels to be the right-shifted inputs. Orange dashed arrows are residual connections. LM head weights are tied to embedding weights. Output hidden states are also sent to the PC module. For SWAG, only an S1 context is used, with 4 possible endings/versions (1 correct, 3 distractors).

3. Methodology

3.1. Story Cloze Test

As noted, the Story Cloze Test was introduced by (Mostafazadeh et al., 2016), and involves choosing the correct single sentence ending to a 4-sentence story. The stories are designed to reflect common, everyday scenarios in simple fashion, while containing causal, temporal relationships.

For multi-task fine-tuning, as seen in Figure 5, the stories' training and testing sets are loaded as inputs, where each sample contains two choices: in each case the first four sentences of the story, followed by either the correct or incorrect ending; a binary label corresponds to which version is correct. A common multiple choice (MC) formulation (Radford et al., 2018) prepends a special start token to each version of the story, with a delimiter token separating the story body from a given ending, and a final classification token is appended, to be used as the representative hidden state (as opposed to using the hidden states for every token in the story versions' sequences). These two hidden states, one for each version of the story, are passed through a linear layer to obtain the MC logits: a pair of scores corresponding to the label, and these two choices' scores are passed through a softmax layer with cross-entropy to obtain the scalar loss against the label. Thus, the loss is the negative log-likelihood (NLL) of the correct choice.

The language modeling objective of the standard SCT task can be written as follows:

$$\mathcal{L}_{LM} = -\sum_i \log P(x_i | x_{i-k}, ..., x_{i-1}; \theta) \tag{1}$$

where θ are the network parameters and k the context window size (Radford et al., 2018).

Next, in the standard paradigm, the auxiliary multiple choice (MC) loss is computed:

$$\mathcal{L}_{MC} = -\sum_{(x,y)} \log P(y | x^1, ..., x^m) \tag{2}$$

where y is the label for which ending is correct, x is the story with start, delimiter, and classification tokens and samples are from the ROCStories dataset used for MTL fine-tuning. Probabilities are determined by softmax, and in the case of Equation 2, the logits or prediction scores for each choice are summaries of the stories' hidden states: that is, they are the hidden states of their respective final classification tokens, as described in §3.1.

3.2. Predictive Coding

We follow the predictive processing inspiration of the above work in §2.1, with some changes. As noted, we use GPT-2 encodings rather than BERT, using sentences' special end tokens as their pooled representations. Our aim is to model the sentences in brief stories from the ROC Stories corpus with a top-down predictive mechanism, with the intent to improve performance on the Story Cloze Test.

Formulations of PC tend to depict updates as based on error information from two sources: the current layer's descending prediction (black arrows in Figure 6) error as fed back to it by the target layer below (e.g. the unidirectional red arrow in Figure 6), and also, via a lateral connection to an error node (the bidirectional red arrow in Figure 6), it receives the error of the layer above that it will pass back upward. As described by Millidge et al. (2021), this use of local error minimization can be viewed as a layer's activity seeking a compromise between the error caused by its deviating from the next layer's prediction and the error it causes through its prediction of the previous layer. This may provide some explanation for the use of a top-down context vector in a layer's context vector computation in Araujo et al. (2021): in lieu of prediction error updates which use the error units as top-down and bottom-up conduits across layers to inform representations, the direct top-down context information may act as a substitute, acting in concert with the pre-existing bottom-up transfer.

The integration of the upper context vector in these previous works comes from the use of an autoregressive (AR) recurrent unit; with GPT-2's causal masking it may be somewhat redundant, although that masking is at the token level and may not capture cohesion between sentences; after early experimentation with a sequence of GRU, fully-connected layer, and activation function, in this work's main experiments we omit sentence-level modeling and simply project GPT-2 sentence representations with a simple linear module to obtain a prediction.

Specifically, we extract 5 sentence representations from each sample, with the first 4 sentences as the story body, and compute the layers' losses for each timestep (each sentence representation serving as condition and target, except the 5th sentence, which serves only as the 4th sentence's target). After averaging this auxiliary loss

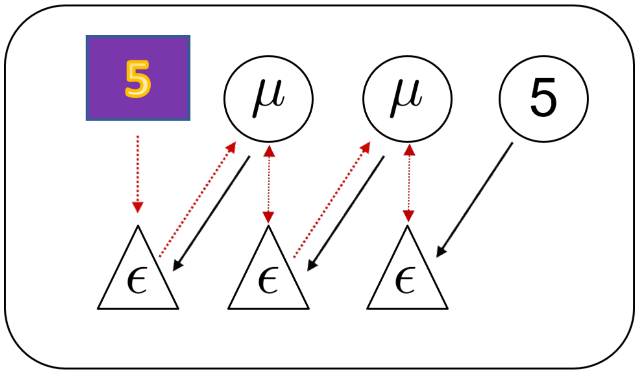

Figure 6: Left to Right: Input image to predicted digit label; recreated from Millidge et al. (2021).

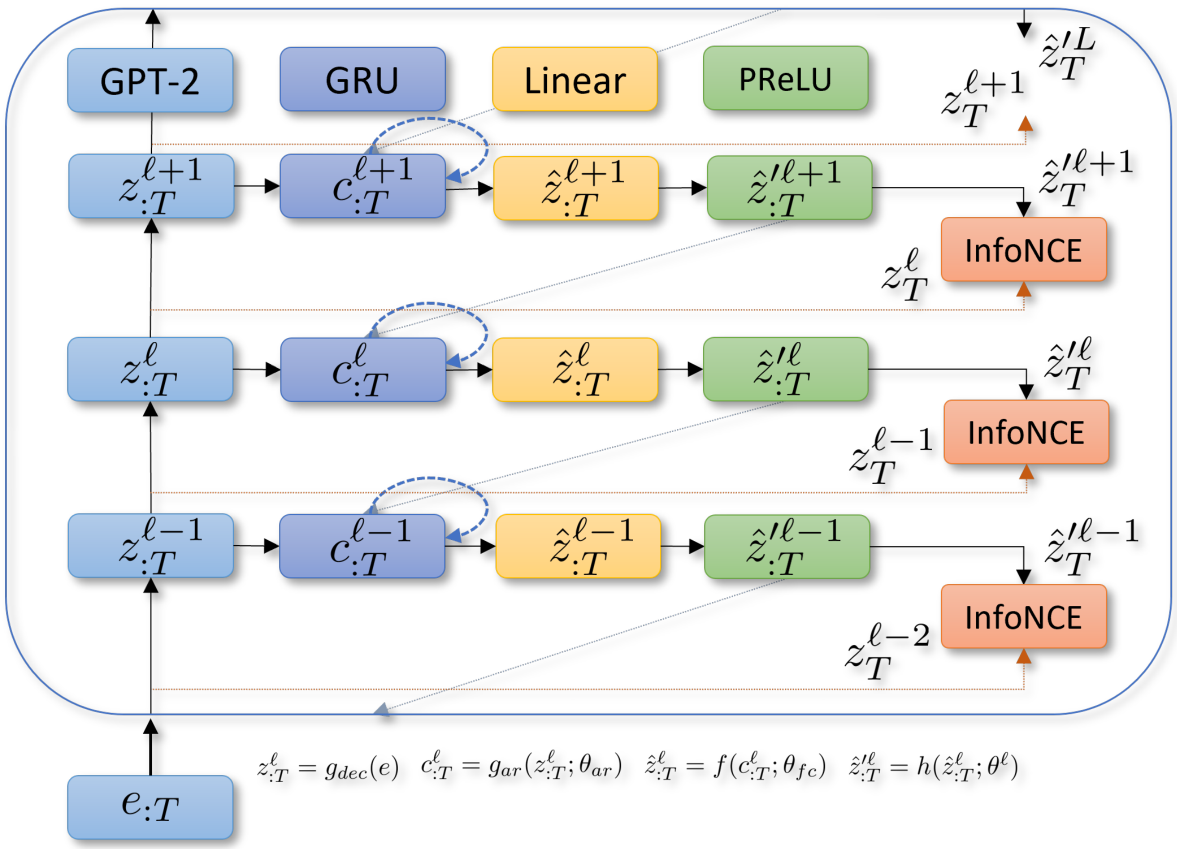

Figure 7: A version of our framework with most configuration options: GPT-2 layers' sentence encodings are passed through GRU, then linear, then parametric ReLU, after which the final sentence representations are used for loss. Optional direct top-down connections would entail concatenating the features of upper layers' context vectors at some point (e.g., after PReLU) to current layers' hidden states' features so that, if GRU is used in the configuration, g_{ar} takes $[\hat{z}^{\ell+1}_{:T}; z^{\ell}_{:T}]$ as input to obtain $c^{\ell}_{:T}$, with down-projection back to original dimensions D ($\theta_{ar} \in \mathbb{R}^{2D\times D}$).

for a batch, it is added to the batch's language modeling loss as originally computed. Specifically in this work, we find it effective to use only the final timestep's layer representations for prediction and loss computation; this use of the final representation will be the default when referencing PC, unless otherwise specified.

In Araujo et al. (2021), given some sequence of sentences and the BERT [CLS] encodings ($z = g_{enc}$) which were produced through the standard feedforward/bottom-up dynamic, an auxiliary loss is computed from the top-down as seen in Figure 2. Proceeding from the final layer encoding down to each previous layer below, a context vector c for each sentence in the sequence (depending on how many sentences are being predicted) is autoregressively obtained from a recurrent neural network g_{ar}, using that layer's [CLS] encodings and the context vector from the layer above, if any: the resulting context vector for a sentence is thereby a function of context vectors up to the current sentence as well as the layer above's context vector to that sentence. The losses are summed across layers and timesteps in order to obtain an auxiliary next-sentence modeling loss. The layer's context vector with its top-down and recurrent information is then used to predict the next sentence representation. In our work, for certain ablations the transformed current layer representation $\hat{z}^{\ell}_{t}$ is compared with the next sentence representation of the previous layer (Figure 7), $z^{\ell-1}_{t+1}$ as the ground truth, slightly different from what is seen in Figure 2. In other ablations, the same timesteps are used across layers (i.e., the target is $z^{\ell-1}_{t}$; Figure 8). In either case, we feel this cascade of downward predictions of incoming input better fits the conceptual framework of hierarchical predictive coding.

$$\mathcal{L}_{PC} = \frac{1}{L}\sum_{\ell=L}^{1}\left(-\frac{1}{M}\sum_{x}^{M}\frac{1}{T}\sum_{t=0}^{T-1}\left[\log\frac{\exp(\frac{\hat{z}^{\ell}_{x,t}\cdot z^{\ell-1\intercal}_{x,t+1}}{\tau})}{\sum_{n}\exp(\frac{\hat{z}^{\ell}_{x,t}\cdot z^{\ell-1\intercal}_{n,t+1}}{\tau})}\right]\right) \tag{3}$$

for M samples in a batch, L layers, and T sentence representations.

As seen in Equation 3, our predictive coding loss $\mathcal{L}_{PC}$ is inspired by the next-sentence modeling loss of Araujo et al. (2021), which in turn was inspired by Oord et al. (2018)'s contrastive predictive coding NCE loss. Equation 3 can be related to the log-bilinear model of Oord et al. (2018), where $\hat{z}^{\ell} \cdot z^{\ell-1\intercal} = z^{\ell}\theta z^{\ell-1\intercal}$. For sample x we take a given sentence representation (a special delimiter token at the end of a sentence) at timestep t for layer ℓ, $z^{\ell}_{x,t}$, and project it to obtain a prediction $\hat{z}^{\ell}_{x,t}$ for the previous layer's actual sentence representation at the next timestep: $z^{\ell-1}_{x,t+1}$. This becomes the positive pair used for InfoNCE, (noise-contrastive estimation: intended to use mutual information to learn 'slow features'–useful features that vary slowly over time Wiskott and Sejnowski (2002)–normalized by negative pairs of $\hat{z}^{\ell}_{x,t}$ matched with other sentence encodings $z^{\ell-1}_{n,t+1}$ taken from $M-1$ target samples $n \neq x$ in the batch.

Rather than predicting a fixed k timesteps ahead as done in the previous works (e.g. Figure 3), we allow for a dynamic k representations until the final token is reached, such that k depends on the current timestep (e.g., sentence 3 predicts sentences 4 and then 5, so that k = 2, k = 3 from sentence 2). In this work we forego this step, as early experiments found it unhelpful.

We also found direct top-down connections with GRU unhelpful: where, to give an example starting with step $t = 2$ and the penultimate layer to help illustrate: the penultimate layer has its second representation c^{L-1}_2 concatenated with c^{L}_2 and projected to obtain $\hat{z}^{L-1}_2$, and the representations $\hat{z}^{L-1}_{:2}$ are passed through a GRU; the result is used to predict z^{L-2}_3 as well as z^{L-2}_4, and z^{L-2}_5, with losses tallied at each instance and then averaged. This is repeated with the other t making predictions in the layer and then the lower layers making their own predictions, and so on until the first hidden layer is reached, predicting the original input embeddings.

As noted and seen in Figure 5, inputs consist of two stories: a plot with the correct ending, and the same plot with the incorrect ending; to avoid redundancy and conflicting signals we use only the plot with the correct ending in our PC loss computation. We surmise that this avoids the redundancy of modeling the story bodies twice, while avoiding potentially mixed signals by the conflicting end-

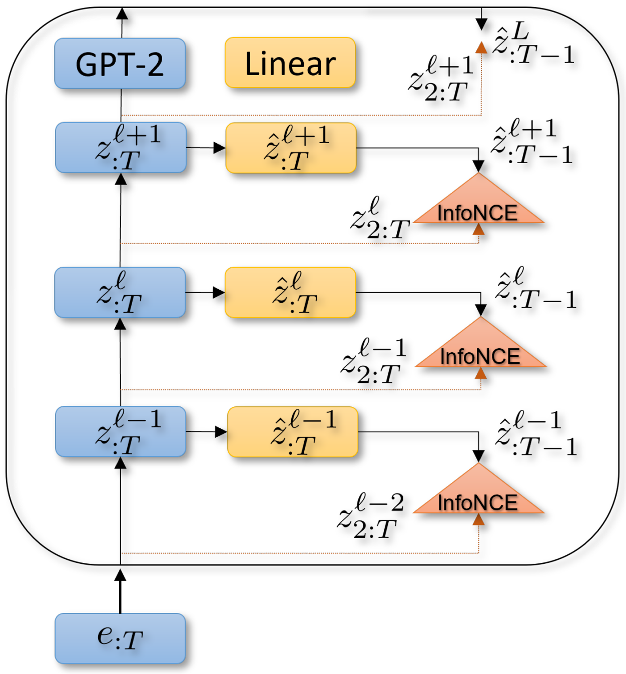

Figure 8: A version of our framework with simpler configuration: GPT-2 layers' sentence encodings $\{1, ..., t, ..., T\}$ are passed through a linear transformation, after which each sentence representation $\forall t \in \{1, ..., T-1\}$ is compared to the next of the previous layer ($t+1 \in \{2, ..., T\}$). Loss is averaged across timesteps at each layer for each sample, and down the layers.

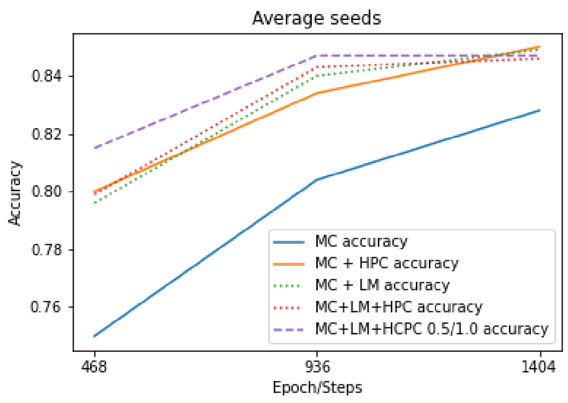

Figure 9: Accuracy on the test set averaged over 5 seeds for all steps. HPC: Hierarchical (layer-wise) PC, last representation at each layer; HCPC: hierarchical with current layer targeting next step of previous layer, illustrated in Figure 8. 0.5/1.0 refers to the coefficients weighting LM and PC losses.

ings, and possibly providing an indirect form of supervision. Previous works which focused on the correct versions include Guan et al. (2020) and Ippolito et al. (2020).

In the baseline, this MC loss $\mathcal{L}_{MC}$ is combined as a weighted sum (Eq. 4) with the language modeling loss $\mathcal{L}_{LM}$, which is the NLL for all of the token predictions in both versions of the story, where vocabulary scores are obtained by using a linear layer to project all of the hidden states output by the Transformer's final decoder block onto the token embeddings.

More generally, the final multi-task loss $\mathcal{L}_{MTL}$ is a sum of main and auxiliary losses with optional coefficients:

$$\mathcal{L}_{MTL} = \mathcal{L}_{MC} + \lambda^{LM}\mathcal{L}_{LM} + \lambda^{PC}\mathcal{L}_{PC} \tag{4}$$

where $\lambda = 1.0$ if not specified.

4. Experiments

4.1. Datasets

The premade SCT datasets consist of a validation and test set with 1871 examples each; following previous works (Cui et al. 2020; Schwartz et al. 2017) we use the validation set for training and evaluate results on the test set. We were unable to create an effective larger training set from the ROCStories corpus, which contains >90k samples and not incorrect endings, by creating incorrect endings to those samples; we suspect distractor endings must be both similar and plausible (i.e., not simply a scrambled version of the correct ending). The latter case that multiple-choice questions feature plausible alternatives may be ideal if the aim is human-like performance, as psychological research has suggested it to be true for human learners (Little et al., 2012).

4.2. SWAG

We train and evaluate models on the SWAG[1] (Zellers et al., 2018) multiple-choice task as well, for one epoch. This task, also intended for grounded commonsense, features shorter contexts with 3 incorrect alternatives, where each context is a caption from a video activity, and correct answers are captions for the next events.

4.3. SST-2

As our approach to PC allows for the use of the same final token used by other classification tasks, we assess the effects of PC loss added to the binary sentiment task with Stanford Sentiment Treebank datasets of movie reviews provided by the GLUE benchmarking suite (Wang et al., 2018).

[1]https://huggingface.co/datasets/swag

4.4. Models

Base: The base GPT-2 model used is the smallest model, considered (Woolf, 2019) to be balanced in terms of performance and speed, with 768 dimensions, 12 layers, 12 attention heads, and 124M parameters.

Baseline (MC): Our baseline is the base GPT-2 model fine-tuned on the SCT task with multiple-choice loss.

Modification (LM+MC): An additional token-level language modeling auxiliary loss is added to the baseline's MC loss, to assess its impact on the standard SCT model.

Modification (PC+MC): An additional hierarchical predictive coding (HPC) auxiliary loss is added to the MC loss; no language modeling objectives during fine-tuning when using only the final step, only a projection of the hidden states in order to contrast with target hidden states.

Modification (LM+PC+MC): Both auxiliary losses are added to the baseline's joint losses, with varying loss weighting coefficients.

4.5. Settings

Hyperparameters are left the same as the original Radford et al. (2018) paper: Adam optimization with epsilon 1e-8, 3 epochs of training, weight decay 0.01, dropout 0.1, and batch sizes reduced to 4 to better fit a single standard GPU such as allotted for free by Google Colab[2]. The model uses default linear learning rate decay, a scheme for scaling the learning rate (starting from a default 5e-5) based on the number of training steps, which is intended to improve stability. Evaluation batch sizes are increased to 12.

The weights of the GRU are initialized with Kaiming normal initialization (He et al., 2015) with parametric ReLU activation (with initial $\alpha = 0.1$ for each layer); linear module weights are default initialized; all PC weights have zero bias initialization, are shared across layers as in Araujo et al. (2021), and are not used at inference. Dynamic padding is used for each batch, to the longest sequence per batch. InfoNCE[3] temperature $\tau = 0.1$. We find that if increasing batch size (e.g. to 8), a concomitant reduction in temperature (e.g. $\tau = 0.05$) seems to maintain performance of InfoNCE loss, perhaps because the normalizing term (Equation 3) is a function of batch size.

4.6. Evaluation

Averaged over 5 seeds[4] and trained for 3 epochs, SCT results are significantly improved over baseline (MC-only) when adding PC loss, competitive with the standard MC+LM formulation (Table 2).

[2]https://colab.research.google.com

[3]https://github.com/RElbers/info-nce-pytorch

[4]A randomly generated sequence of integers: [2175, 5765, 6782, 6966, 7367]

Metric	MC			MC+PC			MC + LM			All (HCPC) λ0.5/1.0		
Epoch	468	936	1404	468	936	1404	468	936	1404	468	936	1404
Acc	0.750	0.804	0.828	0.800	0.834	0.850	0.796	0.840	0.849	0.815	0.847	0.847
F1	0.744	0.799	0.824	0.796	0.830	0.846	0.791	0.835	0.844	0.810	0.843	0.843

Table 2: SCT accuracy and binary F1 on the test set averaged over 5 seeds for all epochs.

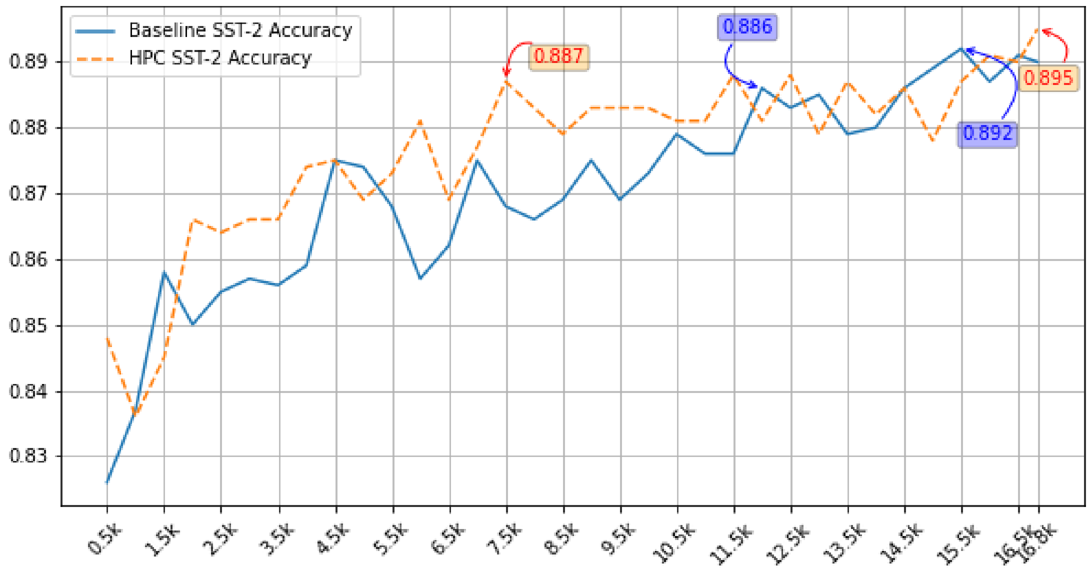

Figure 10: SST-2 accuracy across 16838 steps (1 epoch) averaged over 5 seeds with next highest values at earlier steps shown.

Metric	MC	MC+PC	MC + LM	MC + LM + HCPC
Epoch	18387	18387	18387	18387
Acc	0.553	0.586	0.647	0.651
F1	0.553	0.586	0.647	0.651

Table 3: SWAG accuracy and weighted F1 on the test set averaged over 5 seeds.

We observe a certain instability in the PC approach such that certain seeds or using more features (e.g., 1024 vs. 768) may cause divergence. However, adding a small dose of LM loss (e.g., 10%) rectifies this, stabilizing models for consistent and often slightly improved results (over other configurations). This careful titration of language modeling as a minimal supplement to predictive coding and multiple-choice losses may be optimal for the SCT task.

For SWAG, despite improvement of MC+PC over MC alone, no improvement over MC+LM was gleaned, which we presume may be due to the 4 possible endings diluting potential supervision effects from PC, as well as the much briefer context used to choose among these endings. For SST-2, while performance after an epoch was not significantly improved, it does appear that the PC loss improved results during earlier steps, perhaps increasing the speed of convergence (Figure 10). Interestingly, these changes occurred despite a short context and without a kind of implicit supervision (by computing loss on a 'correct' set of hidden states, as done in the multiple choice SCT and SWAG tasks).

5. Conclusions and Future Work

Here we described the augmentation of GPT-2 with an experimental hybrid predictive coding mechanism, assessing the impact on a downstream NLP task, the Story Cloze Test, with additional application to SWAG and SST-2. It appears that for particular cases, PC can improve results, but outcomes are unstable and require a degree of tuning for different models and tasks. Specifically, it seems that adding predictive coding to the SCT binary multiple-choice task enables models to more effectively distinguish the correct ending, without autoregressive summarizing of sentence-level representations, and is able to do so with both layerwise and temporal predictions of latent representations. For the more difficult MC task required by SWAG, PC only seems to serve as a stabilizing factor for MC loss, while for binary sentiment analysis, the layerwise use of PC may marginally impact the rate of convergence.

A more nuanced investigation is required to understand the impact of predictive coding mechanisms in combination with standard backpropagation, such as examining the coherence of generated endings or other open-ended applications; this study focused simply on standard downstream tasks and straightforward automated metrics. Additional ablations and assessments should more fully specify how representations may be affected, whether harmed or improved, as our preliminary results on the many possible variations are informally observed. Our conjecture is that the mechanism of action is a pressuring of representations to preserve elements of these representations in previous layers of the hierarchy (increasingly low-level components) useful for the task at hand.

References

Abdullahi Ali, Nasir Ahmad, Elgar de Groot, Marcel AJ van Gerven, and Tim C Kietzmann. Predictive coding is a consequence of energy efficiency in recurrent neural networks. *bioRxiv*, 2021. [§2.1.]

Vladimir Araujo, Andrés Villa, Marcelo Mendoza, Marie-Francine Moens, and Alvaro Soto. Augmenting BERT-style models with predictive coding to improve discourse-level representations. *arXiv preprint arXiv:2109.04602*, 2021. [§§1, 2, 2.1, 2.1, 3.2, 3.2, 3.2, and 4.5.]

Zheng Cai, Lifu Tu, and Kevin Gimpel. Pay attention to the ending: Strong neural baselines for the ROC Story Cloze task. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 616–622, 2017. [§2.2.]

Yu-An Chung and James R. Glass. Generative Pre-Training for Speech with Autoregressive Predictive Coding. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3497–3501, 2020. [§2.1.]

Yiming Cui, Wanxiang Che, Weinan Zhang, Ting Liu, Shijin Wang, and Guoping Hu. Discriminative Sentence Modeling for Story Ending Prediction. *ArXiv*, abs/1912.09008, 2020. [§4.1.]

Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minlie Huang. A knowledge-enhanced pretraining model for commonsense story generation. *arXiv preprint arXiv:2001.05139*, 2020. [§3.2.]

Kaiming He, Xian gyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. [§4.5.]

Micha Heilbron, Kristijan Armeni, Jan-Mathijs Schoffelen, Peter Hagoort, and Floris P de Lange. A hierarchy of linguistic predictions during natural language comprehension. *bioRxiv*, pages 2020–12, 2021. [§2.1.]

Geoffrey E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11:428–434, 2007. [§2.1.]

Matin Hosseini and Anthony Maida. Hierarchical Predictive Coding Models in a Deep-Learning Framework. *arXiv preprint arXiv:2005.03230*, 2020. [§§2.1 and 2.1.]

Daphne Ippolito, David Grangier, Douglas Eck, and Chris Callison-Burch. Toward better storylines with sentence-level language models. *arXiv preprint arXiv:2005.05255*, 2020. [§3.2.]

Jeri Little, Elizabeth Ligon Bjork, Robert A. Bjork, and Genna Angello. Multiple-Choice Tests Exonerated, at Least of Some Charges. *Psychological Science*, 23:1337 – 1344, 2012. [§4.1.]

William Lotter, Gabriel Kreiman, and David Cox. Deep predictive coding networks for video prediction and unsupervised learning. *arXiv preprint arXiv:1605.08104*, 2016. [§§2.1, 2.1, and 4.]

Gary Lupyan and Andy Clark. Words and the world: Predictive coding and the language-perception-cognition interface. *Current Directions in Psychological Science*, 24(4):279–284, 2015. [§1.]

Beren Millidge, Alexander Tschantz, and Christopher L Buckley. Predictive coding approximates backprop along arbitrary computation graphs. *arXiv preprint arXiv:2006.04182*, 2020. [§2.1.]

Beren Millidge, Anil Seth, and Christopher L Buckley. Predictive coding: a theoretical and experimental review. *arXiv preprint arXiv:2107.12979*, 2021. [§§2.1, 3.2, and 6.]

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and evaluation framework for deeper understanding of commonsense stories. *arXiv preprint arXiv:1604.01696*, 2016. [§§1, 2.2, and 3.1.]

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. [§§2.1, 3, and 3.2.]

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. [§§1, 2.2, 3.1, 3.1, and 4.5.]

Rajesh PN Rao and Dana H Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87, 1999. [§§2.1 and 2.1.]

Martin Schrimpf, Idan Blank, Greta Tuckute, Carina Kauf, Eghbal A Hosseini, Nancy Kanwisher, Joshua Tenenbaum, and Evelina Fedorenko. The neural architecture of language: Integrative reverse-engineering converges on a model for predictive processing. *BioRxiv*, 2020. [§§1 and 2.1.]

Roy Schwartz, Maarten Sap, Ioannis Konstas, Leila Zilles, Yejin Choi, and Noah A. Smith. The Effect of Different Writing Tasks on Linguistic Style: A Case Study of the ROC Story Cloze Task. *ArXiv*, abs/1702.01841, 2017. [§4.1.]

Yosef Singer, Ben D. B. Willmore, Andrew J. King, and Nicol S. Harper. Hierarchical temporal prediction captures motion processing from retina to higher visual cortex. *bioRxiv*, 2019. [§2.1.]

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. *arXiv preprint arXiv:1905.05950*, 2019. [§1.]

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *ArXiv*, abs/1804.07461, 2018. [§4.3.]

James CR Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262, 2017. [§1.]

Laurenz Wiskott and Terrence J Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002. [§3.2.]

Max Woolf. How to make custom AI-generated text with GPT-2, Sep 2019. URL `https://minimaxir.com/2019/09/howto-gpt2/`. [§4.4.]

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *EMNLP*, 2018. [§4.2.]