Probing for Emergent Relative Word Representation in GPT-2

Julian Baldwin and **Ryan Newkirk**
Northwestern University

Abstract

This paper investigates emergent relative word representation in GPT-2, motivated by the challenges posed by token representations in transformers. Inspired by Neel Nanda's informal exploration and the general project of mechanistic interpretability, this project extends beyond toy inputs to deal with the challenge of probing on natural, cohesive sentences. We provide visualizations and detailed training statistics on linear probing to illustrate the abstract representations of information in GPT-2.

1 Introduction

The transformer architecture has demonstrated remarkable performance in Natural Language Processing (NLP) tasks. A reasonable intuitive understanding of why they work so well is that through layers of self-attention and MLPs (Multi-Layer Perceptrons), they are able to construct and use more and more abstract representations of the raw underlying tokens and corresponding embeddings, which enables their impressive performance.

However, the token representation used in transformers presents certain challenges. Asking models to attend to previous words or to look at corresponding words in different sentences can be difficult, as tokens are often misaligned with words, and large or infrequent words may be split into many fragments.

This project, *Probing for Emergent Relative Positional Embeddings in GPT-2*, explores the hypothesis that the initial layers of the model focus on constructing a representation of a token's position in the sentence, rather than relying solely on the positional embedding provided. For example, the model might develop a representation akin to "This token is the 3rd word in the sentence," as opposed to "This token is in position 5," which is directly given by the positional embedding. This question is directly inspired by Neel Nanda's informal exploration (Nanda, 2023), which demonstrated that probes appear to learn this feature on toy inputs (but, in his own words, "take all this with a mountain of salt").

We first recreate Nanda's results, then extend the probing of the model's embeddings to a dataset of natural sentences. We chose to study GPT-2, as it is publicly available, small enough to run with a single GPU, and has been a popular choice for interpretability studies in the past.

2 Background/Previous Work

The rise of large language models (LLMs) has brought new attention to the field of AI interpretability, particularly through the lens of mechanistic interpretability. This subfield aims to dissect and comprehend the intricate, low-level behaviors of such models. Chris Olah draws an analogy between this goal of reverse engineering of deep learning models and decompiling complex software's binary code, to reveal the underlying mechanisms at play (Olah, 2022). Mechanistic interpretability is young, and often confined to informal explorations of models published as blog posts (like the post that inspired this project!), but has seen academic attention like the notable isolation of a specific circuit in GPT-2 for indirect object identification (Wang et al., 2022).

Probing has emerged as a prevalent method for uncovering the representations learned by various layers within a model (Belinkov, 2022; Gurnee et al., 2023; Hewitt and Manning, 2019). For instance, studies involving the toy task of predicting legal moves in the board game Othello have utilized probes (both linear and non-linear) to decipher the complex representations of an entire board state (Li et al., 2023; Nanda et al., 2023).

This project builds on these efforts by employing straightforward linear probing on residual stream activations in GPT-2. By adopting an extremely simple probing technique, we hope to ensure that the discovered representations are inherent to the

model, rather than an artifact constructed by the probing process. This method is consistent with the broad goals of mechanistic interpretability: to construct a meaningful understanding of how neural networks operate, based on tangible, low-level observations.

3 Methodology

3.1 Dataset

We used various types of text corpora for our project. First, in order to replicate Neel Nanda's original experiments we utilized a dictionary "Random Words" of the 10,000 most common English words (Price, 2022) for random sequence generation. We also utilized multiple text corpora with full coherent sentences:

- "Brown" - Brown Corpus (Voigt, 2024)
- "President" - A corpus of presidential speeches (Voigt, 2024)

To prepare the data for training our model, we needed to generate suitable datasets from these corpora. For our "Random Words" dataset, we used a fixed prefix appended with random sequences of 10 words from the common English corpus. This was an easy to construct initial test without nuances like punctuation. For our full sentence datasets, we fed the model straightfoward sequences from the corpora, without modification.

3.2 Tools

The GPT-2 tokenizer frequently splits long or infrequent words into multiple tokens, so to eventually train probes on relative word position we needed to track each token's index and associated word within each sequence. We used the following tools:

spaCy: A widely used open-source Python package for language processing. Specifically, we used spaCy's *'en_core_web_sm'* model, a lightweight version of the spaCy package which includes a sentence tokenizer (not to be confused with GPT-2's tokens, this identifies and splits words from sentences), and part of speech tagger.

TransformerLens: An open source Python library designed to simplify mechanistic interpretability research for popular transformer models. Specifically, it allows us to load the publicly available weights for GPT-2, inference the model on text samples, and cache the internal activations for training probes. TransformerLens also contains the GPT-2 tokenizer, which splits the raw strings into chunks (such as "tokenizer" -> "token", "izer") and maps them to indices so they can be processed by the transformer model. We wrote a matching algorithm (see Appendix A) to match the GPT-2 tokens and indices to the words isolated by spaCy.

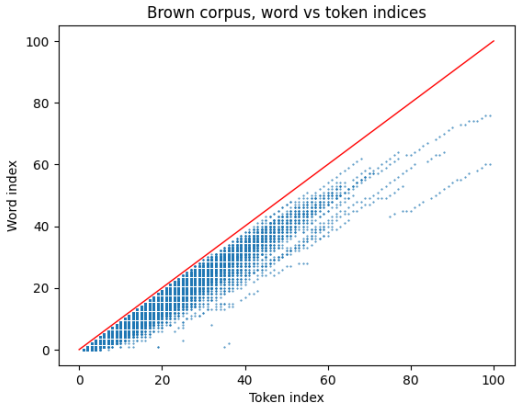

Figure 1: Scatterplot showing the correlation between word and token indices for the full Brown corpus.

3.3 Embeddings

Each sequence (either a string of prefix + random sequence, or a string of multiple coherent sentences) is fed into GPT-2 using TransformerLens, and we specifically save the residual stream activations after each layer of the model. GPT-2 Small has 12 layers, giving 12 possible locations within the model to extract embeddings. The hidden dimension of GPT2-small is 768, so this gives us a 768-dimensional embedding for each token in the input sequence. Each embedding becomes a training sample for our probe, with the label being the word index associated with that token.

3.4 Probes

As mentioned above, we employed the simplest possible probe: a single linear layer. Our probe is implemented using PyTorch (Paszke et al., 2019), and are set up to predict multiple classes, each corresponding to a different word position within the input sequence. This makes the weights of a probe a single matrix with dimensions $(num\_classes, embedding\_size)$.

Training details:

- Trained using cross-entropy loss and the Adam optimizer
- Learning rate of 1e-3
- Weight decay of 1e-3

We did not attempt to tune hyperparameters. After initial testing with probes on our "Random Words" dataset, as show in Figure 2, we chose to

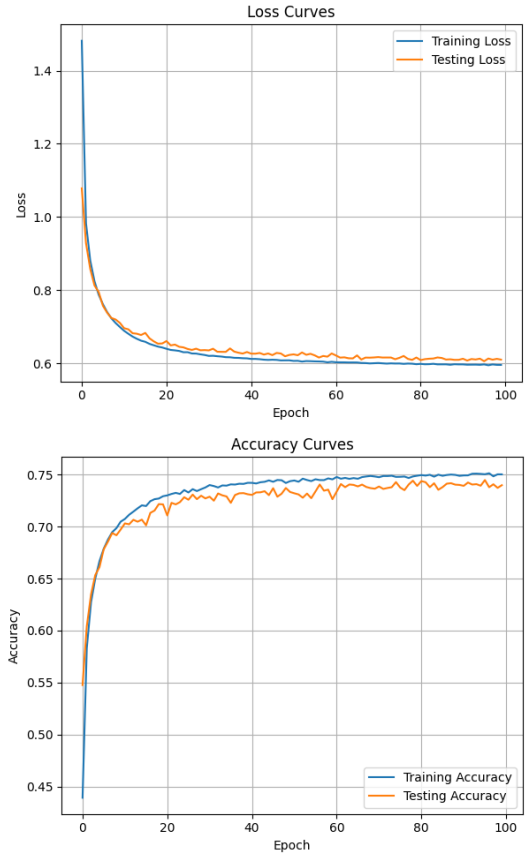

Figure 2: Loss and accuracy curves for linear probe, trained over 100 epochs. Performance on test set peaks after roughly 40 epochs.

Class	Precision	Recall	F1-score
0	0.9912	0.9959	0.9935
1	0.9620	0.9671	0.9646
2	0.9077	0.8955	0.9016
3	0.8185	0.8199	0.8192
4	0.7570	0.6697	0.7107
5	0.6058	0.7790	0.6815
6	0.6451	0.4498	0.5300
7	0.5346	0.7044	0.6079
8	0.5679	0.4098	0.4761
9	0.7340	0.8009	0.7660
		Accuracy	0.7492
		Total Samples	126,209

Table 1: Probe performance on "Random Words" dataset across classes. Each class label corresponds to word index. Probe was trained on layer 3 embeddings.

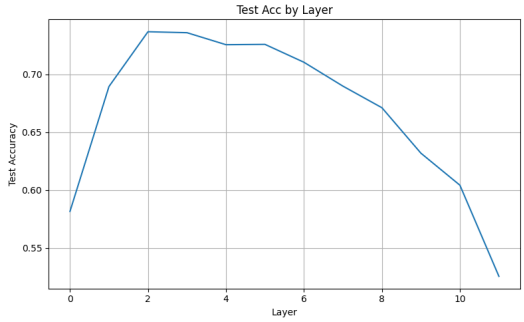

Figure 3: Probe accuracy per layer on "Random Words" dataset.

train probes for a maximum of 40 epochs.

4 Results

4.1 Initial Recreation of Nanda's Results

Table 1 shows performance by class for a probe trained on the "Random Words" dataset, which is a direct recreation of Nanda's experiment. We achieve roughly 75% accuracy, which is strong on a 10-classification task where random guessing would give 10%. However, there is clear correlation between the token index and word index, and the model is given a positional embedding with each token from which the absolute token position can be inferred. Additionally, these inputs always have a fixed prefix and do not have punctuation or coherence between words, meaning the connection is tenuous between this probe's results and the models function while processing typical text.

However, we can still gain some insight into the model's overall structure. Figure 3 shows the accuracy of probes trained on embeddings from each layer of the model, with accuracy peaking in layers 3 and 4 before gradually falling off at later layers. Intuitively, it makes sense that a low level abstraction like word position would appear in early layers, and be less prevalent in the final layers immediately before the model's next token prediction.

4.2 Training on Coherent Sentences

4.2.1 Preparing Dataset

When inferencing the model with full sentences, our previous strategy for classification labels becomes more complicated. Sentences can vary widely in length, so our distribution is now strongly right-skewed (see Figure 4). Additionally, since the longest sentences in our corpus are over 100 words long, labeling each exact word index for every token is unwieldy.

It seems unlikely that the model is attending to the relative position of the 82nd and 83rd words as closely as to the 2nd and 3rd words, so we chose to classify on a percentile basis. We take each indi-

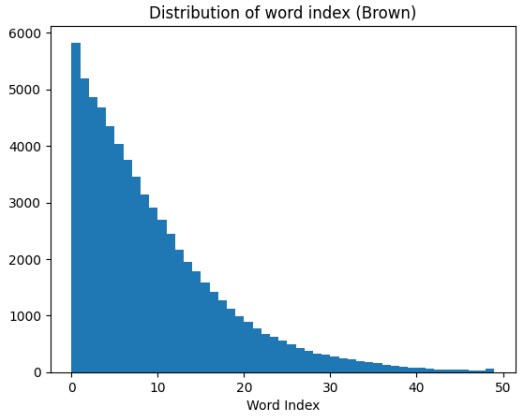

Figure 4: Distribution of word index for the Brown corpus.

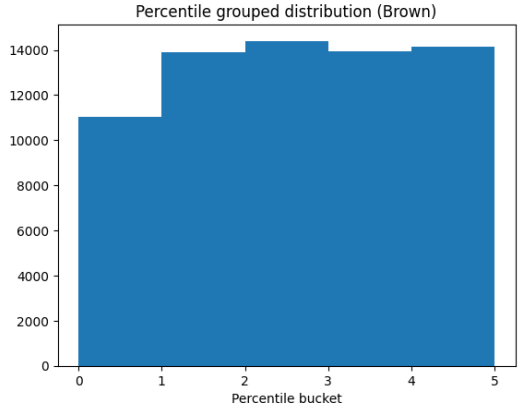

Figure 5: Distribution of Brown corpus after grouping into quantiles.

vidual corpus, calculate quantiles for the corpus's distribution of word indices, and then group each into a bucket with a distinct label. This flattens the distribution of labels (see Figure 5, and allows us to moderate the number of labels to find the most natural target for the probe. This still reflects meaningful positional information about each token, so it still matches with our overall goal.

4.2.2 Baselines

Our baseline is a probe trained on a randomly initialized model. It gets roughly 10% accuracy, nearly identical to random guessing, showing that a linear probe is not a strong enough model for predicting relative word position training on random noise. This supports our core assumption that the GPT-2 model is creating structure from the data in an interpretable way.

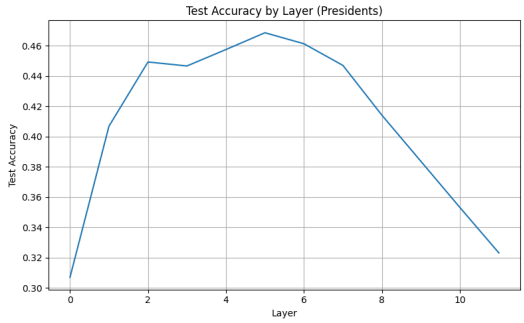

Figure 6: Probe accuracy per layer on "President" dataset.

4.2.3 Evaluating probes

Table 1 shows performance across many different training schemes for the natural sentence datasets. Training with embeddings from GPT-2 Medium (which has more layers and a slightly larger hidden dimension of 1024) gives a slight improvement in performance. All metrics are from probes trained on the residual stream after layer 3, based on initial testing. However, this may not have been optimal. Figure 6 shows accuracy on President dataset from probes trained across all layers of GPT-2 Small, which diverges from the pattern displayed on the simple "Random Words" dataset. Here, performance peaks solidly in the middle layers. This difference could be attributed to how the model processes longer, more coherent sequences, or due to the change in how labels are generated. Perhaps representing a word's quantised position rather than straight word index within a sentence requires more layers of processing.

4.3 Visualization

4.3.1 Probe Directions

In Figure 7 we can see how distinct the different prediction directions are. Some notable points:

- Class 2 is nearly opposite of Class 7 and 8
- Class 0 is closer to the Class 9 than it is to Class 1-3
- The final classes form a square-shaped area of similarity, which might indicate the model has trouble differentiating the nuances of word positions in the latter part of a sentence.

Figure 8, layers in the middle have broadly the same direction, though it clearly gradually shifts throughout layers. Doesn't really show signs of being a consistent direction throughout the entire model. The first and last layer's direction is the most distinct, which makes sense given their

Model	Dataset	Samples	Bins	Accuracy	Baseline
GPT-2 Small	President	425k	20	0.3022	0.05
GPT-2 Small	Brown	366k	10	0.6276	0.1
GPT-2 Small	Brown	366k	5	0.7945	0.2
GPT-2 Small	President	425k	10	0.4484	0.1
GPT-2 Small	President	425k	5	0.6387	0.2
GPT-2 Medium	Brown	366k	10	0.6283	0.1
GPT-2 Medium	Brown	366k	5	0.8025	0.2
GPT-2 Medium	President	425k	10	0.4856	0.1
GPT-2 Medium	President	425k	5	0.7044	0.2

Table 2: Probe performance on full sentence datasets. Various bin sizes for computing labels. Baseline shows rounded accuracy of a probe trained on embeddings from randomly initialized model.

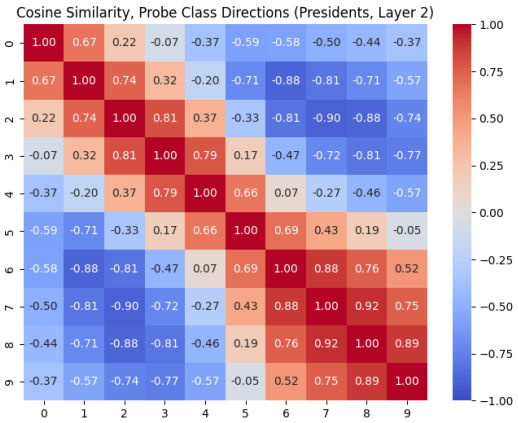

Figure 7: Cosine similarities between learned directions of probe trained on layer 2 residual stream, "President" dataset

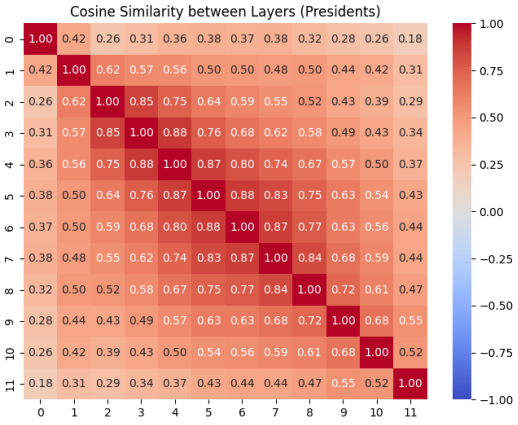

Figure 8: Cosine similarities for the same class, between probes trained on different layers, "President" dataset

As I walked through the park arm in arm with George R. R. Martin, he told me that he did not know I was with him in the video. When I asked him, "Would you rather see my daughter or my husband or his mother" or "do you want to see anything that is too dangerous for you to talk about?" I knew that I was giving him that speech. But there was the same thing I knew, not knowing if I was doing the right thing. I knew that I was doing what I was called when I was with George R. R. Martin in the library because I would have it translated to a better country.

Figure 9: Projection from embedding to Layer 2's probe direction 0 (start of sentence, first few words). Text was generated by GPT-2

probe's poor accuracy.

4.3.2 Text Examples

In Figure 9, we can see the result of projecting onto a single probe direction. Note how the probability is higher at the beginning of each sentence, but it isn't simply responding to each period, like in "George R. R. Martin".

From there, using a baseline of one of our best performing models (10 bins, GPT-2-medium, trained on the President dataset), we can see comparing Figures 10 and 11, the model using 10 bins qualitatively seems to have a better performance of projecting higher values to words at the start of sentences. Comparing Figures 10 and 12, we can see the projection is less consistent in magnitude for GPT-2-small, and comparing Figures 10 and 13, we notice similar projections, where the Brown corpus seems to be more precise in it's projections but also mistakes numerical digits for beginnings of sentences, which may be due to the dataset having less numbers than the President dataset.

5 Discussion

From our probing experiments, both in a simplified setting with random sequences of words and using cohesive sentences from popular text corpora, we see clear evidence of GPT-2 forming emergent representations of the relative word position of each token (considering either absolute index or quantiles). This is a minor feature in the grand scheme of all of GPT-2's capabilities, but the ability of linear probes to localize these representations to the early and middle layers are revealing of the overall structure of how transformers gradually compute and reshuffle abstractions in order to generate realistic text. We hope this investigation can be another small piece on the road to a more mechanistic understanding, and eventually steering, of transformers.

Some future potential work:

- Probe for emergent positional representations at different levels of syntax, like a word's position within a paragraph, or a sentence's position within a paragraph.
- It seems reasonable that the model, instead of having an internal linear direction for each word index or quantile, has a more regression-based representation of word position. Framing our probing as a regression task instead of a classification task could be a useful follow up.
- Investigate if model can identify a word's part of speech in a sentence, or relative position for specific parts of speech (e.g. "is this token the 2nd name in the sentence?" which could be useful for the model's performance on something like indirect object identification).
- Performing interventional experiments using the directions learned by probes. As previous studies have demonstrated (Belinkov, 2022; Li et al., 2023), interveneing on or ablating the model's activations during inference can be useful in determining whether there is a causal relationship between a model's representations, or if it is just an artifact of the probing process.
- Investigating the impact of including or excluding punctuation from sentences on the performance of the model, and seeing how that might effect the tokenization process

Group Contribution

Julian had the initial idea for the project, wrote the project proposal, and did the initial coding to recreate Neel Nanda's results. We then both worked getting the full sentence corpora to work for training probes, and both worked on creating visualizations (Ryan created the highlighted text samples, Julian created the heatmaps).

We both contributed equally to writing the report itself. GPT4 converted our Python code into a Latex algorithm block (see Appendix A)

Ethics Statement

We do not believe this work has any ethical considerations. No models were harmed in the making of this paper.

Acknowledgements

thx to rob voigt for good class

References

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023. Finding neurons in a haystack: Case studies with sparse probing.

John Hewitt and Christopher D. Manning. 2019. 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)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Emergent world representations: Exploring a sequence model trained on a synthetic task.

Neel Nanda. 2023. Tiny Mech Interp Projects: Emergent Positional Embeddings of Words. https://www.lesswrong.com/posts/Ln7D2aYgmPgjhpEeA/tiny-mech-interp-projects-emergent-positional-embeddings (Accessed: 13 March 2024).

Neel Nanda, Andrew Lee, and Martin Wattenberg. 2023. Emergent linear representations in world models of self-supervised sequence models.

Chris Olah. 2022. Mechanistic interpretability, variables, and the importance of interpretable

bases. https://www.transformer-circuits.pub/2022/mech-interp-essay.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library.

Eric Price. 2022. 10,000 most common english words. https://www.mit.edu/~ecprice/wordlist.10000.

Rob Voigt. 2024. Assignment 2 - ngram lm. https://faculty.wcas.northwestern.edu/robvoigt/courses/2024_winter/ling334/assignments/a2.html. (Accessed: 13 March 2024).

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small.

A Appendix A: Token Matching Algorithm

We cannot strongly guarantee that this algorithm will function correctly on all Unicode-strings, but we have verified that it gives reasonable looking results for our test corpora.

B Appendix B: More Text Examples

Algorithm 1 Match GPT-2 Tokens to spaCy Word Indices

Input: text ▷ Input text to process
Output: mappings ▷ List of tuples with token, its index, and word index
tokens ← GPT2_Tokenize(text)
tokenStrings ← ConvertTokensToStrings(tokens)
spaCyWords ← spaCy_Tokenize(text)
Initialize tokenIndex, wordIndex, subWordIndex to 0
Initialize mappings to []
while tokenIndex < Length(tokenStrings) **do**
    token ← tokenStrings[tokenIndex].strip()
    if not token **then** ▷ Skip whitespace tokens
        AddMapping(mappings, token, tokenIndex, -1)
        Increment tokenIndex
        continue
    end if
    currentWord ← spaCyWords[wordIndex]
    if token is part of currentWord starting at subWordIndex **then**
        AddMapping(mappings, token, tokenIndex, wordIndex)
        Increment subWordIndex by Length(token)
        Increment tokenIndex
    else
        if NextWordExists(wordIndex, spaCyWords) **and** token in NextWord(spaCyWords, wordIndex) **then**
            Increment wordIndex
            Reset subWordIndex to 0
        else
            AddMapping(mappings, token, tokenIndex, -1)
            Increment tokenIndex
        end if
    end if
end while
return mappings

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Figure 10: Projections from embedding to Layer 2's probe direction 0 from a model using GPT-2-medium, 10 bins, and trained on the President dataset.

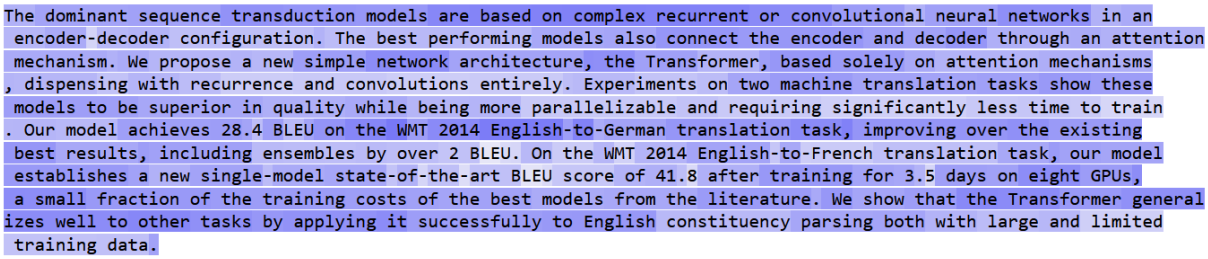

Figure 11: Projections from embedding to Layer 2's probe direction 0 from a model using GPT-2-medium, **5 bins**, and trained on the President dataset.

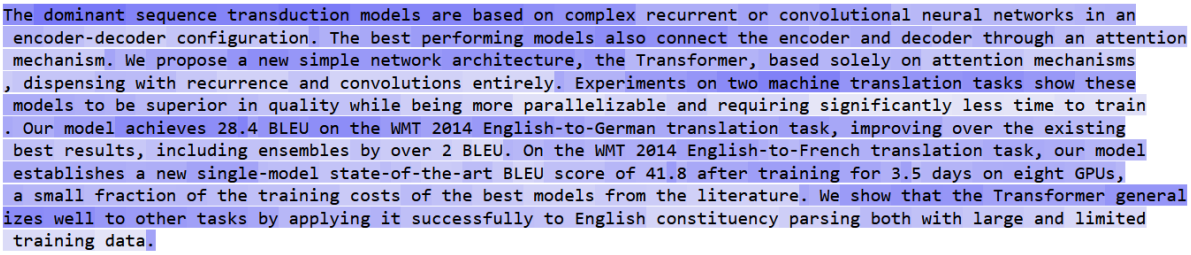

Figure 12: Projections from embedding to Layer 2's probe direction 0 from a model using **GPT-2-small**, 10 bins, and trained on the President dataset.

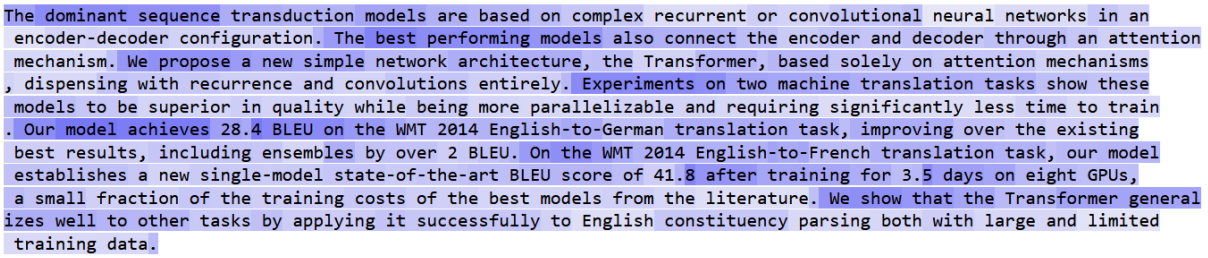

Figure 13: Projections from embedding to Layer 2's probe direction 0 from a model using GPT-2-medium, 10 bins, and trained on the **Brown Corpus**.

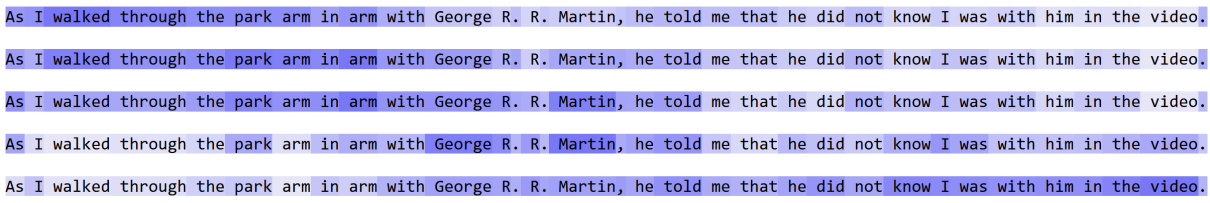

Figure 14: Projection onto each probe direction (5 percentile buckets) on the same text example.

```
def map_word_index(text, prepend_bos=False):
    gpt_tokens = model.to_tokens(text, prepend_bos=prepend_bos).squeeze(0)
    gpt_tokens_str = [model.to_single_str_token(int(t)) for t in gpt_tokens]

    doc = spacy_model(str(text))
    word_idxs = [t.text for t in doc if t.is_alpha]

    if not len(word_idxs):
        return torch.Tensor(), []

    i = 0
    cur = 0 # current word index.
    sub_idx = 0 # sub index of current word.

    result = []
    while i < len(gpt_tokens_str):
        t = gpt_tokens_str[i].strip()
        # skip if token is just a newline or other whitespace.
        if not len(t):
            result.append((t, i, -1))
            i += 1
            continue

        cur_word = word_idxs[cur]
        # if token is part of current word, update sub_idx, continue to next token.
        if cur_word.find(t, sub_idx) != -1:
            result.append((t, i, cur))
            sub_idx += len(t)
            i += 1
        else:
            # if token not in cur_word, check
next word.
            if cur+1 < len(word_idxs) and t in word_idxs[cur+1]:
                cur += 1
                sub_idx = 0
            # if not in cur_word or next word, give up and continue.
            else:
                result.append((t, i, -1))
                i += 1

    return gpt_tokens, result
```

Figure 15: Projection onto each probe direction (5 percentile buckets) on an input of code (our matching algorithm in Appendix A).

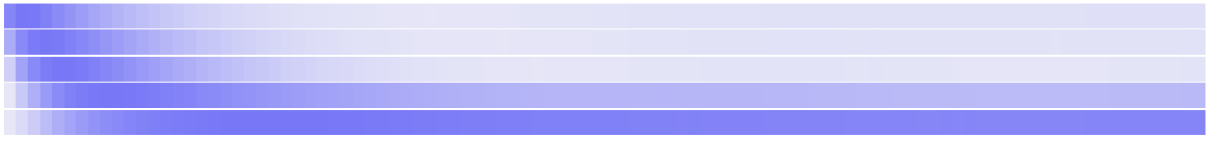

Figure 16: Projection onto each probe direction (5 percentile buckets) on an input of all spaces.

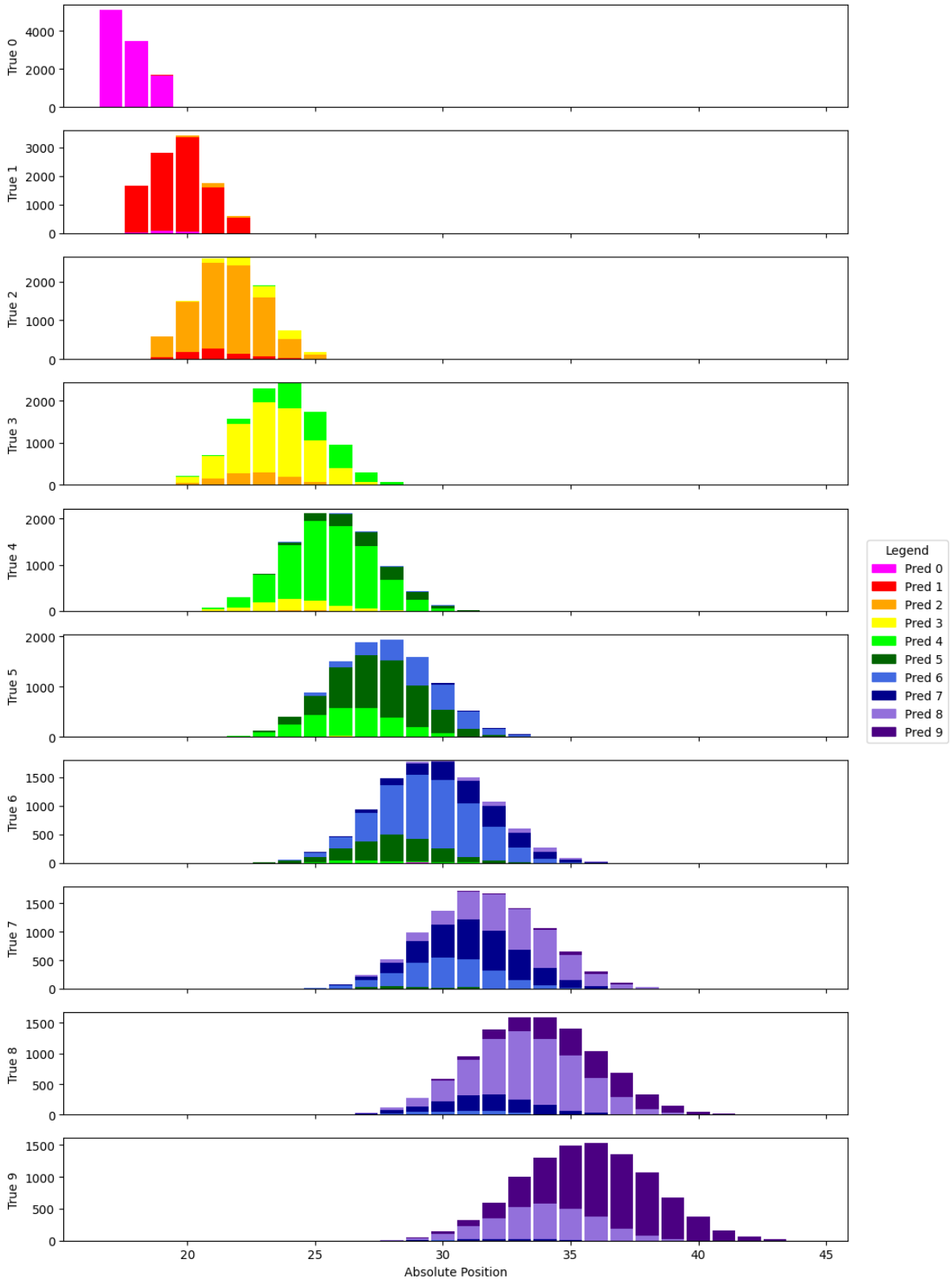

Figure 17: This plot is a recreation of Nanda's key plot in his informal exploration (Nanda, 2023). It illustrates the absolute distribution of model predictions for word indices against absolute token positions in the prompt, showcasing varying levels of accuracy and error across different indices and positions.