Probe-less Probing of BERT's Layer-Wise Linguistic Knowledge with Masked Word Prediction

Rithvik Reddy Sama and Akash Mehta
Department of Electronics and Computer Engineering
University of Cincinnati
Cincinnati, OH 45221, USA

samary@mail.uc.edu , mehta2ah@mail.uc

Abstract

Language models like BERT have shown remarkable skill in capturing complex linguistic patterns, yet understanding the specifics of this knowledge remains an intriguing and challenging endeavor. Our study delves into the reimplementation of BERT's ability to comprehend language by analyzing its masked word prediction on an English corpus. We implement a computationally efficient novel approach termed Probe-less Probing, which uses masked word prediction to directly interrogate BERT's layer-wise linguistic knowledge, without any explicit probing used traditionally. We also investigate how the accuracy changes layerwise. The STREUSLE 4.4 dataset, encompassing a wide variety of lexical-semantic and syntactic information, is the dataset for this study.

In the first phase of the project, our task involved generating masked word predictions and examining their accuracy. In the second phase, we will have to examine the accuracy across different layers and compute differential scores to quantify the relative contribution of each layer to the prediction accuracy, both in terms of the word itself and its part of speech (POS).

Keywords—Bert, Probe-less Probing, STREUSLE 4.4, Masked-Word prediction, Linguistic knowledge

I. INTRODUCTION

The BERT model also known as Bidirectional Transformers for Language Understanding developed by GOOGLE in 2018 was originally proposed by Devlin [1]. The specialty of the BERT model is that it can process the text in both directions simultaneously which in turn helps to capture the contexts of words which helps the model to understand better. The BERT model by [1] was built on the transformer architecture which comprises of transformer block, embedded layers, residual layers, and output layers. BERT model is pre-trained on large text corpora using two unsupervised tasks:

- Masked Language Model- Random sentences are masked and fed to the neural network for the prediction of these masked tokens.
- Next Sentence Prediction: A pair of sentences are given and the model should predict whether two sentences are consecutive or not.

GOOGLE has divided the BERT model into two types:

- Bert-base consists of 12 layers of encoder.
- Bert-large consists of 24 encoding layers.

The BERT model has significantly been a very impactful language model demonstrating superior performance in incorporating contextual information in word representations[6]. It excels in a wide range of various downstream tasks compared to state-of-the-art benchmarks. How much information is encoded in its contextual word representations across layers is an intriguing topic chosen for this project.

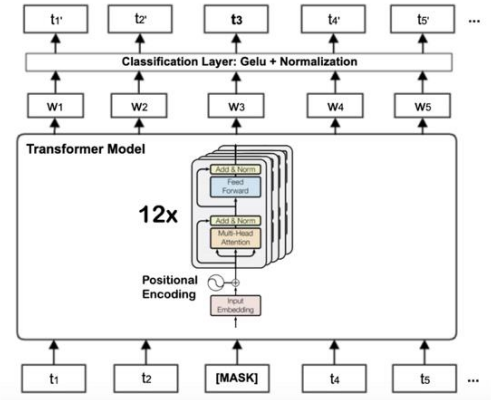

Figure 1: Bert-base 12 layers

The BERT's layers we reimplemented (12 layers in bert-base). Traditional methods used to study use external probing classifiers which utilize the internal structure of the model or behavior of the model at inference. While structure-based probes have successfully linked specific domains of linguistic knowledge to local regions, the reliance on probing classifiers raises questions about whether such knowledge is genuinely encoded or merely an artifact of the classifier. Behavior-based probes, on the other hand, forgo external classifiers but often focus on qualitative analyses of final layer outputs, leaving quantitative analysis of layer-wise outputs unexplored.

While re-implementing the original study[2], we adopt a behavioral probing approach to explore layer localization. Specifically, we employ token masking to observe how BERT predicts masked tokens. By analyzing whether BERT predicts the same word, a word with the same part of speech, or neither, we can calculate and discern whether BERT can predict masked tokens or POS-tagging better.

II. DATASET

STREUSLE 4.4: We used the STREUSLE 4.4 dataset used in the original study (Aoyama and Schneider, 2022). STREUSLE[5] stands for Supersense-Tagged Repository of English with Unified Semantics for Lexical Expressions. It is a corpus of web reviews written in English which has 723 documents, 3,813 sentences, and 55,590 tokens, and is laden with ample annotations of numerous lexical-semantic and syntactic information. There are 17 attributes in total for each token and each as its importance. The attributes and their importance are given in the image below.

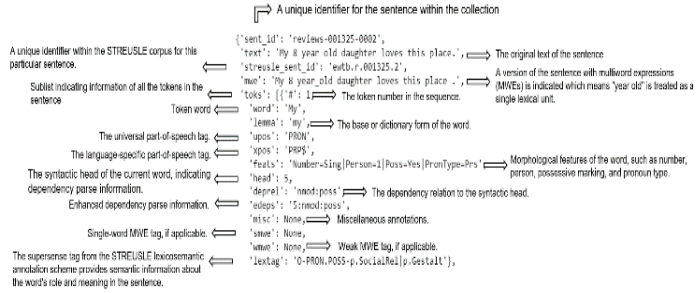

Figure 2: Dataset

For this project reimplementation according to the paper we choose 4 attributes ‘upos’, ‘word’, ‘toks’, ‘mwe’. ‘upos’ stands for universal part-of-speech which refers to the consistent part-of-speech tagging conventions that apply universally across different languages. The ‘upos’ consists of ‘NOUN’ refers noun, ‘VERB’ refers verbs, ‘ADJ’ refers adjectives, ‘ADV’ refers adverbs, ‘PRON’ refers pronouns, ‘DET’ refers determiners and articles, ‘ADP’ refer adpositions, ‘NUM’ refers numerals, ‘CONJ’ refers conjunctions, ‘PRT’ refers particles, ‘INTJ’ refers interjections, ‘SYM’ refers symbols, ‘X’ refers other, ‘PUNCT’ refers punctuations. These tags are designed to be simple yet sufficient for use in a wide range of linguistic tasks. ‘word’ refers to the token word extracted from the sentences of the dataset. ‘toks’ refers to the token number in the sequence. ‘mwe’ refers to the Multi-Word Expressions are the phrases that won't make sense when they are fully deductible from the meaning of the individual words.

III. METHODOLOGY

We initiated our analysis by loading the STREUSLE 4.4 corpus, into a Google Colab. For ease of handling, we converted the streusle.conllulex file to streusle.json format[5]. Subsequently, loading the pre-trained BERT-base-uncased (12 layers) model from Hugging Face's Transformers library. We then followed the preprocessing of the data by the most used methods which are Segmentation and tokenization[3].

1. We create ‘n’ variants where n is considered as the length of individual tokenized sentences, at each variant the subsequent word is masked manually, and in simple terms, we replace the word with [MASK].

2. Once all the data is masked using the [MASK] token, we run it with vanilla BERT.

3. We ran it with the vanilla BERT and let predict the [MASK] token that was earlier replaced in step 1. We repeated across each layer and for all the masked version sentences.

4. After predicting we used the library stanza[7] and extracted ‘en’ and used this library for extracting the part of speech of the predicted masked word token and then replaced the token [MASK] with the predicted word and then detokenized.

5. We then look for the multiword expressions for the predicted token word, if any it returns all the ‘mwe’ words if not returns NONE.

6. Once we ran it with all the masked tokenized sentences by the end, we extracted the original sentence, predicted sentence, original token, predicted token, original pos, and predicted pos.

For better understanding a flowchart is given below:

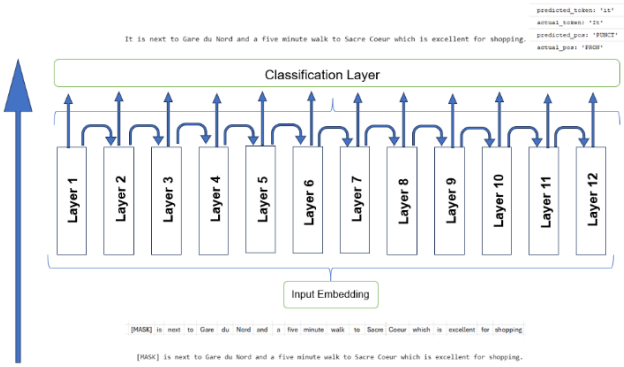

Figure 3: Visualization of BERT across its layers

We then conducted an analysis categorized by syntactic category and presence in multiword expressions (MWEs), employing POS tagging to categorize predictions, and examining both the accuracy of word predictions and their syntactic categories. Text data is tokenized using BERT's tokenizer[3]. This involves converting text into tokens that BERT can process. The tokenizer also adds special tokens like [CLS] at the beginning and [SEP] at the end of sentences. For each masked token prediction, hidden states from all layers of BERT are extracted. These hidden states represent the model's internal understanding and features at various levels of abstraction.

To find out how well our model performs, the predicted tokens are compared against the original tokens and their part-of-speech (POS) tags to compute accuracy. The accuracy is measured both at the token level (whether the predicted token matches the original token) and at the POS level (whether the predicted token's POS matches the original POS). For word prediction accuracy, we predicted masked words in sentences extracted from the dataset. The accuracy was calculated by comparing the predicted words with the ground truth words in the original sentences. We got an accuracy of 0.4327 which means the bert-base model when provided a sentence with a masked word, correctly predicts the missing word 43.27% of the time. For POS tagging, we used the same bert-base model to predict the POS tags of masked words. The accuracy was calculated by comparing the predicted POS tags with the actual POS tags of the words in the original sentences. We got an accuracy of 0.6708 which means the bert-base model correctly identified the POS tag 67.08% of the time. The methodology includes a layer-wise analysis to understand at which layer of the Transformer model the most accurate predictions occur. This involves calculating the prediction accuracies for each layer separately and observing how accuracy changes across layers.

Similarly, we calculate the expected layer for every UPOS based on the POS, POSM, WORD, and WORDM. POS is the expected layer for POS matching for general tokens[4]. The value represents the average layer at which the model's prediction matches the actual POS of the masked word. In POSM, the "M" stands for Multiword Expression. This column shows the expected layer for POS matching but specifically for tokens that are part of multiword expressions. The expectation is calculated in the same way as the POS column but filtered for MWE instances. WORD, like the POS column, represents the expected layer for exact word matching for general tokens. WORDM focuses on tokens that are part of MWEs, showing the expected layer for exact word matching within that subset. The "Expected Layer" is calculated using a formula that weighs each layer by the differential gain observed from the previous layer, essentially providing an average "center of mass" for where significant changes in prediction accuracy occur. The differences highlighted in TABLE 1 support deeper insights into how syntactic and lexical information is processed differently depending on the complexity and type of the linguistic structure involved, thereby informing about the layer-wise encoding dynamics of the BERT model.

IV. Equations

Analyzing the descriptive statistics, to quantify the relative contribution of each layer to POS match and word match. Deferential scores[4] at each layer are calculated using:

$$\Delta_T^{(l)} = Score_T^{(l)} - Score_T^{(l-1)}$$

POS match and WORD match which in simple language we can as the accuracy score of the POS and the accuracy score of the word are calculated using the above provided incremental gain formula. Once the WORD match and POS match using the incremental gain formula is calculated now the expected layer is calculated in which all the gain scores across the 12 different layers are calculated using the below formula:

$$E_\Delta[l] = \frac{\sum_{l=1}^{L} l \cdot \Delta_T^{(l)}}{\sum_{l=1}^{L} \Delta_T^{(l)}}$$

If the differential scores across the different layers were uniformly distributed, then the expected layer would simply be 6. If the contribution of higher layers were higher than the expected layer would be greater than 6, and vice versa.

Results and Comparison

The accuracy of the word, pos and only pos across the 12 layers of the Bert-base model is given below:

The accuracy for the only pos tends to increase linearly between 0 – 2 layers, from 2 – 10 the accuracy tends to increase slowly, and from 10 – 12 the change in accuracy seems to be very slow. The difference in the accuracy of pos and word match is very minute, and it increases linearly from the lowest layer to the highest layer. The dotted line across the graphs depicts the expected layer which means the particular layer of the model can perform very well as compared with the rest of the layers. The expected layer for pos is between 6 and 6.5 and so as for the word. The expected layer for the only pos is between 3.75 – 4.

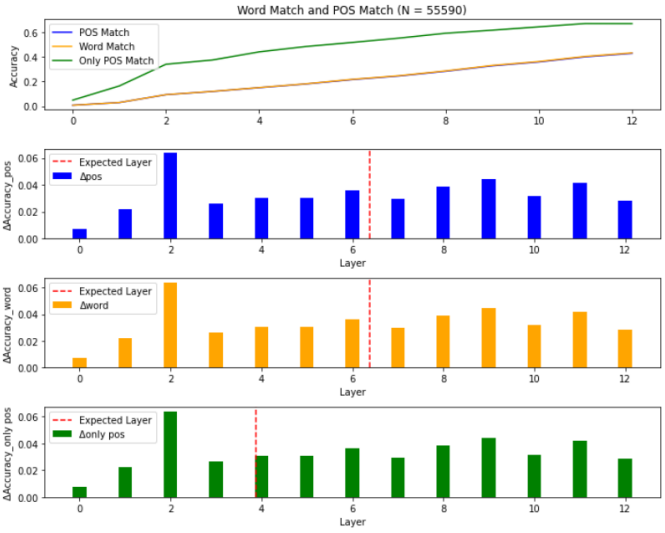

Figure 4: Accuracy across the different layers and Expected layer(Ours)

UPOS	N	POS	POSm	WORD	WORDm
ADJ	3169	7.80	7.89	8.70	8.94
ADV	3080	5.98	5.85	7.29	8.31
INTJ	108	7.60	0.00	8.18	10.25
NOUN	7265	7.25	7.21	9.11	9.01
PROPN	1406	6.73	6.85	8.63	8.59s
VERB	5328	6.27	6.36	8.42	8.65
ADP	3368	6.07	6.01	7.38	7.84
AUX	2950	6.79	6.76	7.62	7.60
CCONJ	1803	5.56	5.49	5.51	5.55
DET	3525	9.25	0.00	6.59	6.60
NUM	555	7.21	8.67	7.66	9.10
PART	1314	6.09	5.77	6.15	5.70
PRON	5264	6.49	6.35	7.82	8.41
SCONJ	808	7.05	7.09	7.79	5.83
X	69	7.18	5.67	8.69	0.00
PUNCT	5958	5.56	3.79	5.90	7.57
SYM	159	7.78	7.75	7.86	8.12

Table 1: Expected Layer by UPOS (Ours)

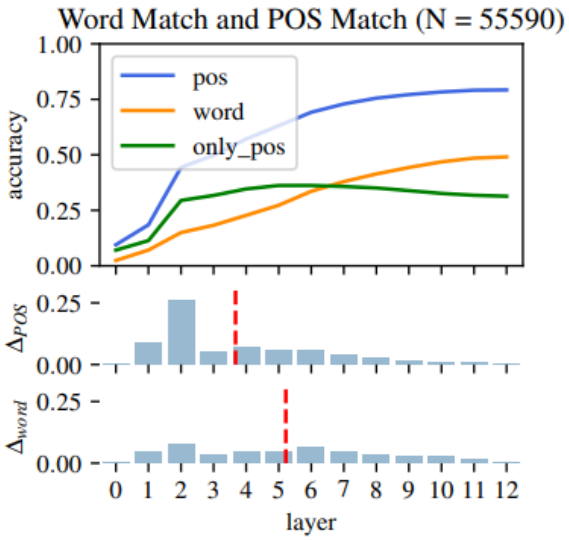

Figure 5: Accuracy across the different layers and Expected layer(Original)

		N	POS	POS_M	word	$word_M$
open	ADJ	3169	4.04	4.24	6.45	6.35
	ADV	3080	3.42	3.76	5.74	5.30
	INTJ	108	3.48	**9.33**	7.13	**8.75**
	NOUN	7265	3.98	4.48	7.53	6.99
	PROPN	1406	**6.68**	6.11	**8.05**	7.88
	VERB	5328	3.96	3.68	6.73	6.38
closed	ADP	3368	3.16	3.52	5.01	5.18
	AUX	2950	3.10	4.43	5.14	5.08
	CCONJ	1803	5.48	5.32	5.88	4.74
	DET	3525	2.16	2.54	3.11	3.43
	NUM	555	5.70	6.81	6.73	7.23
	PART	1314	1.80	1.31	2.08	1.40
	PRON	5264	3.91	4.61	6.76	5.96
	SCONJ	808	5.05	4.45	5.71	5.45

Table 2: Expected Layer by UPOS (Ours)

Figure 4 and 5 compare the Accuracy across the different layers and expected layer[2]. Both graphs show that POS matching accuracy increases with the depth of the layers. This indicates that as the BERT model processes the input through successive layers, it becomes better at predicting the correct part of speech for masked words. This pattern holds consistently in both the original and the reproduced figures. Word match accuracy also increases with layer depth, but it starts to plateau in the higher layers. This suggests that while early layers are better at picking up on contextual cues for word prediction, the improvements in accuracy taper off as you go deeper into the network. Again, this trend is evident in both figures. With only POS match accuracy, this measure accounts for instances where the POS is correctly predicted, but the exact word is not. The initial layers have lower accuracy for only POS match, with a significant increase in middle layers, which then flattens out or slightly decreases. This might indicate that middle layers are specialized in understanding the grammatical structure of the sentence rather than the specific lexical content. There appears to be a slight shift in the layer at which the highest POS match accuracy is achieved. In the reproduced results, the peak accuracy seems to occur at a slightly earlier layer than in the original results. This could suggest subtle differences in model behavior or training. It's also possible that the reproduction used a different version of BERT, spaCy or stanza[7], which could cause such a shift. The difference in accuracy between layers (Δ_pos and Δ_word) can provide insights into which layers contribute most to the increase in accuracy. The original figure shows a more pronounced peak in Δ_pos in the lower layers, suggesting that initial layers contribute significantly to POS understanding. The reproduced results show a smoother transition, with less pronounced peaks, indicating a more gradual increase in POS matching across layers. In the reproduced results, the final layers do not show as much of a decline in Δ_word as in the original results. This could imply that in the reproduced experiment, the model does not overfit specific words in the higher layers to the same extent, or it could be an artifact of the different datasets or random initialization used in the reproduction.

Table 1 and Table 2 compare the expected layer activations for different Universal Parts of Speech (UPOS) tags between the results in the paper and those you've obtained from your code. The two tables show results for metrics such as POS match accuracy (POS), POS match accuracy in multiword expressions (POSm), word match accuracy (word), and word match accuracy in multiword expressions (wordM).

Our POS accuracy is generally higher across most UPOS categories compared to the paper. For example, in ADJ, our POS accuracy is 7.80, while the paper reports 4.04. This could indicate that our model is better at predicting the correct part of speech, or it might be a sign of overfitting or a difference in the evaluation method. In some cases, our POSm values are significantly higher, such as in NUM (your value is 8.67 compared to 2.16 in the paper) or DET (your value is 9.25 compared to 2.54 in the paper). These results suggest a very different performance on multiword expressions, which is quite striking.

The word match accuracy in your results is also generally higher. For instance, the NOUN category shows a word accuracy of 7.25 in your results, whereas the paper reports only 3.92. The discrepancies in wordM, the word accuracy for multiword expressions, are again very notable. Our NOUN wordM accuracy is 9.01 compared to 6.99 in the paper.

V. Conclusion

Reasons for the discrepancies:

- Updates in the model or tokenizer: If the model or tokenizer has been updated since the paper was written, even if the model name is the same, these updates could influence the results.
- Model configuration: Even with the same model name, configuration options such as the number of layers, hidden units, or attention heads can affect results.
- Different random seeds: Even with the same setup, neural networks can yield different results due to the randomness in weight initialization and data shuffling.

Furthermore, we have 3 extra UPOS categories, specifically X, PUNCT and SYM. The tag X often represents a token that does not fit into any other category, and if such tokens were never part of MWEs in the entire 69 instances, then that could explain the expected layer being 0.00 for wordm. The same goes for INT where POSm has 0 instances out of the whole population of 108, Perhaps there were no instances of interjections in MWEs, or they were not captured due to some reason.

Acknowledgment

We would like to thank our Natural Language Processing professor, Dr. Jiang Tianyu, for his constant insights and feedback. His guidance helped us throughout this process.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding.

[2] Tatsuya Aoyama, Nathan Schneider. 2022. Probe-Less Probing of BERT's Layer-Wise Linguistic Knowledge with Masked Word Prediction.

[3] Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties.

[4] Yoav Goldberg. 2019. Assessing bert's syntactic abilities.

[5] John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations.

[6] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language?

[7] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages.