CogBERT: Cognition-Guided Pre-trained Language Models

Xiao Ding, Bowen Chen, Li Du, Bing Qin, Ting Liu
Harbin Institute of Technology
{xding, bwchen, ldu, qbing, tliu}@ir.hit.edu.cn

Abstract

We study the problem of integrating cognitive language processing signals (e.g., eye-tracking or EEG data) into pre-trained models like BERT. Existing methods typically fine-tune pre-trained models on cognitive data, ignoring the semantic gap between the texts and cognitive signals. To fill the gap, we propose CogBERT, a framework that can induce fine-grained cognitive features from cognitive data and incorporate cognitive features into BERT by adaptively adjusting the weight of cognitive features for different NLP tasks. Extensive experiments show that: (1) Cognition-guided pre-trained models can consistently perform better than basic pre-trained models on ten NLP tasks. (2) Different cognitive features contribute differently to different NLP tasks. Based on this observation, we give a fine-grained explanation of why cognitive data is helpful for NLP. (3) Different transformer layers of pre-trained models should encode different cognitive features, with word-level cognitive features at the bottom and semantic-level cognitive features at the top. (4) Attention visualization demonstrates that CogBERT can align with human gaze patterns and improves its natural language understanding ability.[1]

1 Introduction

Pre-trained models such as BERT (Devlin et al., 2019), GPT (Radford and Narasimhan, 2018) and RoBERTa (Liu et al., 2019) have brought promising improvements to natural language processing (NLP) tasks, such as event prediction (Li et al., 2019) and sentiment analysis (Hoang et al., 2019).

On the other hand, from a language processing perspective, cognitive neuroscience studies the biological and cognitive processes underlying language processing in human brains. Researchers specifically designed pre-trained models to capture how the brain represents language meaning

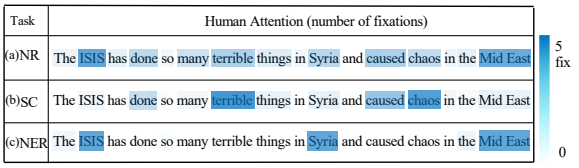

Figure 1: Human attention in different settings. NR (Normal Reading) means human reading without tasks.

(Schwartz et al., 2019; Toneva and Wehbe, 2019). Prior work mainly incorporates cognitive signals by explicitly fine-tuning the pre-trained model to predict language-induced brain recordings.

However, this line of work cannot give a fine-grained analysis and interpretation of why cognitive data is helpful for NLP. This is of great importance to guide future cognition-inspired NLP studies on what kind of cognitive features should be induced from cognitive data and how these cognitive features contribute to NLP tasks. For example, Figure 1 shows eye-tracking data from native speakers of English, where Figure 1 (a) illustrates the number of fixations during the normal reading of humans. Figure 1 (b) and (c) show the number of fixations given the NLP task of sentiment classification (SC) and named entity recognition (NER), respectively. We can observe that human attention is different for the same sentence, given different NLP tasks. In particular, for the SC task, people pay more attention to emotion words, such as "*terrible*" and "*chaos*". While for the NER task, people tend to focus on named entity words, such as "*ISIS*" and "*Syria*". Unfortunately, prior studies cannot give such fine-grained analysis by simply fine-tuning pre-trained models on cognitive data.

To facilitate this, we propose CogBERT, a cognition-guided pre-trained model. Specifically, we focus on the effects of using eye-tracking data, which provides gaze information from native speakers by tracking eye movements and measuring fixation duration. Instead of directly fine-tuning BERT

[1]Our code will be released upon publication.

Proceedings of the 29th International Conference on Computational Linguistics, pages 3210–3225
October 12–17, 2022.

Name	Independent	Strands	Index
Word Position	✓	Lower	1
Word Length	✓	Lower	2
NER Word	✓	Lower	3
Content Word	✓	Lower	4
Noun Phrase	✓	Upper	5
Emotion Word	✓	Upper	6
Active/Passive Aux Poss/Prep Mod Coordinating Conj	✓	Upper	7
Prep/Adj Comp Prep Obj	✓	Upper	8

Table 1: Cognitive feature set. For syntactic features, to avoid sparsity issue, we group active and passive aux, poss and prep mod, and coordinating conj as one feature. Prep, adj comp, and prep obj are grouped as one feature. Aux, Poss, Prep, Mod, Conj, Adj, Comp, Obj are abbreviations for auxiliary, possession, preposition, modifier, conjunction, adjective, complement and objective.

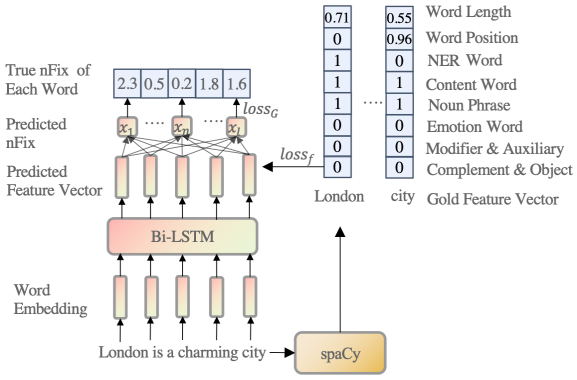

Figure 2: Feature prediction by the Bi-LSTM.

on cognitive data, we first extract psycholinguistic features based on cognition theory (Scarborough et al., 2009). Then we filter out statistically insignificant features in the eye-tracking data (which means that the human attention of words with these features is not significantly higher/lower than the average attention of words). Subsequently, we incorporate these cognition-validated features into BERT by fine-tuning on different NLP tasks. In the fine-tuning process, we would learn different weights for each type of feature according to different NLP tasks.

Our results show that CogBERT can perform better than baseline systems on three benchmark datasets across ten NLP tasks. We give a detailed quantitative analysis of the contributions of different cognitive features to different NLP tasks. Case studies show that CogBERT does learn patterns of human reading behavior compared to BERT. Our findings can provide more insights into cognition-enhanced NLP.

2 CogBERT

CogBERT works in a two-stage setting. The first stage is to induce cognitive features from texts and assign weights for these features guided by human reading signals. In the second stage, we integrate cognitive features into BERT and learn task-specific feature weights for different NLP tasks.

2.1 Cognitive Features Induction and Validation

Psycholinguistic studies (Scarborough et al., 2009) indicate that the acquisition of human reading ability is reflected in two aspects: "*lower strands*" and "*upper strands*". The lower strands (including phonology, morphology, etc.) work together as the reader becomes accurate and automatic with repetition and practice. Meanwhile, the upper strands (including language structures, semantics, etc.) reinforce each other and weave together with the lower strands to produce a skilled reader. Inspired by previous work, we construct an initial cognitive feature set including 46 fine-grained cognitive features extracted from texts using the spaCy tool (Honnibal and Montani, 2017) and divide them into lower strands features (word-level) and upper strands features (semantic/syntax-level).

However, not all cognitive features are statistically significant in eye-tracking data (Zuco 1.0 (Hollenstein et al., 2018), Zuco 2.0 (Hollenstein et al., 2020), and Geco (Cop et al., 2016))[2]. Hence, we filter out cognitive features that the human attention of words with these features is not significantly higher/lower than the average attention of words, retaining 14 usable cognitive features and group them as 8 independent features shown in Table 1. The statistically significant values of each cognitive feature and details of feature validation are shown in Appendix Table 1.

2.2 Learning Weighted Cognitive Feature Vector

We can extract features from texts by using the spaCy tool. Nevertheless, these features should not be assigned to the same or random weights, as their contributions to fitting human understanding of sentences are different. Hence, as shown in Figure 2, given an input sentence $S = \{w_1, w_2, \ldots, w_l\}$ with l words, we train a four-layer Bi-LSTM to

[2]Eye-tracking data is preprocessed by averaging across readers.

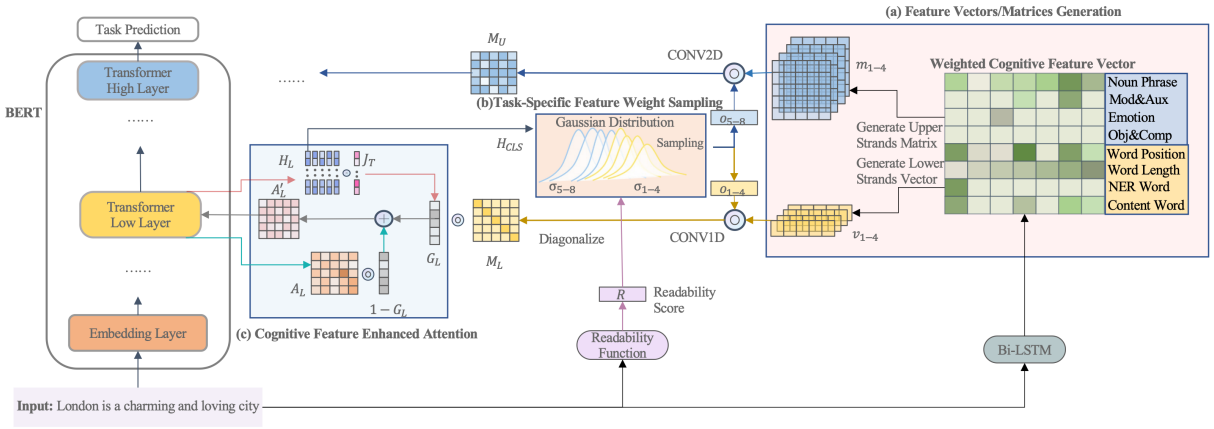

Figure 3: CogBERT Architecture. The Bi-LSTM generates feature vectors, which are used to generate feature matrices $\mathbf{m}$ and feature vectors $\mathbf{v}$. The readability score R and embeddings of [CLS] tokens H_{CLS} are used to compute feature weights o.

map each word embedding to a weighted eight-dimensional cognitive feature vector. According to the cognitive theory (Scarborough et al., 2009), we believe that cognitive features can explain the allocation of human gaze information. Hence, we use gaze information (the number of fixations, nFix) of eye-tracking data (Zuco 1.0, Zuco 2.0, and Geco) as the supervision signals to train the Bi-LSTM model (We also use fixation duration as the supervision signals and achieve the same experimental results). We use the Mean Square Error (MSE) loss $loss_G = \frac{1}{l}\sum_{i=1}^{l}(x_i - y_i)^2$, where $x \in \mathbb{R}^l$ is the predicted nFix score and $y \in \mathbb{R}^l$ is the golden feature score.

On the other hand, to avoid predicting an unreasonable feature score, we also compute the MSE between the predicted feature score matrix $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_l\} \in \mathbb{R}^{l \times r}$ and the golden score matrix $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_l\} \in \mathbb{R}^{l \times r}$ (for example, if a word w_i is a NER word, its golden feature score is 1 on the feature dimension of NER), where $\mathbf{p}_i$ and $\mathbf{q}_i$ is a r-dimensional vector and r is the number of features, which we denote as loss $loss_F = \frac{1}{l}\sum_{i=1}^{l}\frac{1}{r}\sum_{j=1}^{r}(p_{ij} - q_{ij})^2$. Then the model is trained with the objective $loss_T = \alpha \times loss_G + (1-\alpha) \times loss_F$, where $\alpha \in [0, 1]$ is a model parameter to weight the two loss functions (the best development results were obtained with $\alpha = 0.5$).

Note that we use the Bi-LSTM model to predict cognitive features rather than using a regression model to compute the feature score. The main reason is that our goal is to learn the human reading behavior rather than simply fitting the data.

2.3 Incorporating Cognitive Features into BERT

2.3.1 Feature Vectors/Matrices Generation

As shown in Figure 3 (a), for each input sentence S with l words, we can obtain its an $l \times r$ feature matrix from the Bi-LSTM model.

• For each lower strands feature (i.e., word length, word position, NER and content word), we can generate an initial l-dimensional feature vector $\mathbf{v}$ for it from the Bi-LSTM model, respectively.

• For each upper strands feature (i.e., NP chunk, emotion word, Mod&Aux and Obj&Comp), we can generate an initial $l \times l$ feature matrix $\mathbf{m}$ for it from the Bi-LSTM model, respectively. If k adjacent words make up an upper strands feature, its value in $\mathbf{m}$ is the average feature score of k adjacent words obtained from the Bi-LSTM model. The rest of values in $\mathbf{m}$ are filled with 0.

2.3.2 Task-Specific Feature Weight Sampling

We argue that different cognitive features should be given different weights when learning different NLP tasks. For example, the emotion word feature should be more important than others for a sentiment classification task. To address this issue, we utilize the Flesch readability assessment (Kincaid et al., 1975) to evaluate the contribution of cognitive features to sentence readability of different NLP tasks. In other words, given a specific task, if a feature can improve the readability of the input sentence, it should be given a higher weight.

Formally, we use the Flesch readability assessment score R and the embedding of [CLS] token H_{CLS} of a layer in BERT to control the weights of different features. Given a sentence S, the number

of words in S is s_w and the number of syllables in S is s_s. Then the readability score of this sentence is calculated as $R(S) = 206.835 - 1.015 s_w - 84.6 \frac{s_s}{s_w}$, where constants (i.e., 206.835, 1.015 and 84.6) in $R(S)$ are empirical values from Kincaid et al. (1975).

Since $R \in \mathbb{R}^1$ is a one-dimensional number, $H_{CLS} \in \mathbb{R}^{n \times 1}$ is a n-dimensional vector ($n = 768$ for BERT-Base) and cognitive features set T contains eight features including $\{\mathbf{v}_1, ..\mathbf{v}_4, \mathbf{m}_1, ..\mathbf{m}_4\}$, where $\mathbf{v}$ represents a feature vector and $\mathbf{m}$ represents a feature matrix, respectively, we first need to map R and H_{CLS} to an eight-dimensional vector O, to assign a reasonable weight for each feature.

In particular, as shown in Figure 3 (b), we map R and H_{CLS} to a variance set $\sigma = \{\sigma_1, \dots \sigma_r\} \in \mathbb{R}^r$ of a normal distribution $B \sim Norm(0, \sigma^2)$, where the mean of B is zero. σ is computed as:

$$\sigma = f(N^T R) \circ f(M^T H_{CLS}) = \{\sigma_1, \dots, \sigma_r\}, \quad (1)$$

where $\circ$ is the Hadamard product and f is the $tanh$ activation function. $N \in \mathbb{R}^{1 \times r}$ is a mapping vector and $M \in \mathbb{R}^{n \times r}$ is a mapping matrix. Then the weighted feature set $T^{'}$ is computed as:

$$\begin{aligned} T^{'} &= T \circ O = \{o_1 \mathbf{v}_1, \dots, o_4 \mathbf{v}_4, o_5 \mathbf{m}_1, \dots, o_r \mathbf{m}_4\}, \\ O &= B(0, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)_{x=0} \\ &= \frac{1}{\sigma\sqrt{2\pi}} = \{o_1, \dots, o_r\}, \end{aligned} \quad (2)$$

where o_i is the ith feature weight sampled from the normal distribution B with sample point $x = 0$.

2.3.3 Cognitive Feature Enhanced Attention

Inspired by the previous work (Jawahar et al., 2019), which indicates that BERT captures surface features in lower layers and semantic features in higher layers, as shown in Figure 3 (c), we incorporate different cognitive features in different layers, where lower strands are in lower layers and upper strands are in higher layers. We use CNN (LeCun et al., 2015) to compute a feature-enhanced attention score for lower strands and upper strands, which we denote as $M_L \in \mathbb{R}^{l \times l}$ and $M_U \in \mathbb{R}^{l \times l}$, respectively,

$$\begin{aligned} M_L &= Diag(1DCNN(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)), \\ M_U &= 2DCNN(\mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3, \mathbf{m}_4), \end{aligned} \quad (3)$$

where $Diag$ refers to filling the output of $1DCNN$ in the diagonal of the feature-enhanced attention matrix M_L, this is because the generated lower strands features by the Bi-LSTM are word-level feature vector rather than a matrix, and $1DCNN$ and $2DCNN$ means 1-dimensional and 2-dimensional CNN network. The kernel size of both $1DCNN$ and $2DCNN$ is 3, respectively. Then we obtain the cognitive feature-enhanced lower layer attention matrix $A'_L \in \mathbb{R}^{l \times l}$ and higher layer attention matrix $A'_U \in \mathbb{R}^{l \times l}$ as:

$$\begin{aligned} A'_L &= M_L \circ G_L + A_L \circ (1 - G_L), \\ A'_U &= M_U \circ G_U + A_U \circ (1 - G_U), \\ G_L &= f(J^T H_L), \\ G_U &= f(J^T H_U), \end{aligned} \quad (4)$$

where $A_L \in \mathbb{R}^{l \times l}$ and $A_U \in \mathbb{R}^{l \times l}$ are the original attention matrix of lower and higher layers, respectively. $G \in \mathbb{R}^{1 \times l}$ is a gated vector for a transformer head, which is used to balance the combination between the original attention and the cognitive feature-enhanced attention. $H = \{H_{CLS}, H_0 \dots, H_l\} \in \mathbb{R}^{n \times l}$ is the hidden state of the input sentence S, in which H_{CLS} is the embedding of [CLS] token of BERT. $J \in \mathbb{R}^{n \times 1}$ is a vector to map H to the gate vector G in a specific head. The $\circ$ is the row-level Hardmard product that each value of G multiplies with each row of the matrix (e.g., M_L), so that the size of the matrix remains the same after multiplication. The process is same for all layers of BERT.

2.4 Training Details

We train the Bi-LSTM model with hidden size 256, dropout ratio 0.15, 300-dimensional GloVe embedding (Pennington et al., 2014) and 40 epochs. In experiment, we fine-tune the CogBERT using AdamW (Loshchilov and Hutter, 2017) optimizer with learning rate from [1e-5, 3e-5, 5e-5], warm-up ratio of 0.1. The training epochs are form [3, 5, 10] and batch size from [16, 32] for GLUE Benchmark. For CoNLL2000 Chunking, we use batch size from [64, 128], learning rate from [5e-6, 1e-5, 5e-5], and epochs from [40, 60].

3 Experiments

To show the effectiveness of CogBERT, we compare it with baselines on three benchmark datasets across ten NLP tasks and an eye-tracking prediction task.

3.1 Baselines

We compare CogBERT with:

Models	SST2	COLA	MRPC	RTE	MNLI(m/mm)	QQP	STS-B	QNLI
BERT-Base	93.5	51.7	87.2	67.2	84.3/83.7	71.1	85.4	90.4
Syntax-BERT-Base	**94.0**	54.1	89.2	68.9	**84.9/84.6**	72.0	86.7	91.1
fMRI-EEG BERT-Base	93.4	52.9	87.4	67.5	84.3/83.8	71.2	85.3	90.5
Eye-tracking BERT-Base	93.3	51.9	87.5	67.3	84.3/83.7	71.2	85.8	90.6
CogBERT-Base (Random)	93.2	51.1	85.4	66.0	84.1/83.2	71.0	85.3	88.6
CogBERT-Base	**94.0**	**55.1**	**89.5**	**69.4**	**84.9/84.6**	**72.1**	**87.2**	**91.3**
BERT-Large	94.9	60.5	89.3	70.1	86.8/85.9	72.1	86.5	92.7
Syntax-BERT-Large	**96.1**	61.9	92.0	74.7	86.7/86.6	72.5	88.5	**92.8**
fMRI-EEG BERT-Large	—	—	—	—	—	—	—	—
Eye-tracking BERT-Large	94.7	60.7	89.2	70.2	86.6/85.7	72.3	86.7	92.5
CogBERT-Large (Random)	94.4	60.1	88.7	69.2	86.3/85.4	71.8	86.3	92.4
CogBERT-Large	**96.1**	**62.1**	**92.1**	**74.9**	**86.8/86.6**	**72.8**	**89.7**	**92.8**
RoBERTa-Base	95.4	57.1	90.8	73.8	86.3/86.2	72.5	87.4	92.2
Syntax-RoBERTa-Base	**96.1**	63.3	91.4	**81.2**	87.8/85.7	73.5	88.3	**94.3**
CogRoBERTa-Base (Random)	95.3	56.8	90.5	73.4	86.1/85.8	72.1	87.2	92.0
CogRoBERTa-Base	95.7	**63.5**	**91.7**	79.3	**88.1/86.2**	**73.8**	**88.5**	93.9
RoBERTa-Large	96.3	63.8	91.0	84.2	89.5/89.7	72.7	90.2	94.2
Syntax-RoBERTa-Large	**96.9**	64.3	92.5	85.0	90.2/90.0	73.1	**91.4**	**94.5**
CogRoBERTa-Large (Random)	96.1	63.6	90.7	83.8	89.2/89.4	72.3	90.0	94.0
CogRoBERTa-Large	96.5	**64.6**	**92.8**	**85.3**	**90.4/90.3**	**73.5**	90.5	**94.5**

Table 2: Results on the test set of GLUE Benchmark. F1 scores are reported for QQP and MRPC. Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. The best results are bold. For the SST2 task, the Emotion feature weight of CogBERT is set as 0.

• **BERT** (Devlin et al., 2019): A transformer-based pre-trained language model achieved SOTA performances on various NLP tasks.

• **Syntax-BERT** (Bai et al., 2021): A syntax-enhanced pre-trained model outperformed conventional pre-trained models on various NLP tasks.

• **fMRI-EEG BERT** (Schwartz et al., 2019): This model is fine-tuned on EEG and fMRI data, which showed improvements in brain activity prediction and achieved competitive performances on various NLP tasks.

• **Eye-tracking BERT**: It fine-tunes BERT on nFix eye-tracking data following the method used in fMRI-EEG BERT.

• **CogBERT (Random)**: The feature weight of CogBERT is randomly generated rather than supervised learning by human reading signals.

3.2 GLUE Benchmark

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) is a set of tasks to test the model's ability to understand natural language. We implement our method in BERT, RoBERTa on its development set. Results are shown in Table 2. Based on the observation of the experiment results, we find that:

(1) Compared with BERT and RoBERTa, CogBERT achieves consistently better results (when compared with RoBERTa, we use RoBERTa as the base pre-trained model of CogBERT) on all tasks from SST2 to QNLI showing the effectiveness of incorporating eye-tracking can be useful for various tasks.

Models	P (%)	R (%)	F1 (%)
BERT-Base-Cased	95.32	95.23	95.33
SeqVat (unsupervised)	95.39	95.47	95.45
SeqVat (supervised)	96.36	96.31	96.34
CogBERT-Base (Random)	94.46	95.03	94.74
CogBERT-Base	**96.48**	**96.59**	**96.54**

Table 3: Results of CoNLL-2000 text chunking. CogBERT is based on BERT-Base-Cased.

(2) Comparison between Syntax-BERT and CogBERT shows that cognitive features can further improve the performance of BERT on NLP tasks. This is mainly because on the one hand cognitive features used in this paper have already included fine-grained syntax structure features. On the other hand, we involved more psycholinguistic features in CogBERT validated by eye-tracking data.

(3) CogBERT outperforms fMRI-EEG BERT and Eye-tracking BERT, which indicates that fine-tuning on cognitive data cannot fully exploit the value of cognitive data. By inducing fine-grained cognitive features from cognitive data can provide a new perspective for this line of work.

(4) Compared to CogBERT (Random), CogBERT achieves consistently better performances, which confirms that different cognitive features contribute differently to language comprehension in brains. Learning weighted cognitive feature vector is effective for CogBERT.

Tasks	Word Length	Content Word	NER	Word Position	Emotion	NP Chunk	Mod & Aux	Comp & Obj
COLA	0.42	0.56	0.59	0.41	0.78	**1.00**	0.83	0.60
MRPC	0.52	0.68	**1.00**	0.45	0.95	0.76	0.88	0.74
RTE	0.54	0.83	0.67	0.52	0.91	**1.00**	0.73	0.64
CoNLL-2000 Chunking	0.54	0.83	0.94	0.52	0.92	**1.00**	0.64	0.93
CoNLL-2003 NER	0.44	0.51	**1.00**	0.49	0.42	0.39	0.49	0.51

Table 4: Results of intrinsic evaluation of CogBERT-base. The numbers are feature weights in different tasks.

3.3 Sequence Labeling

In addition to the GLUE benchmark, we also evaluate CogBERT on the CoNLL-2000 dataset (Sang and Buchholz, 2000) for text chunking and the CoNLL-2003 dataset (Sang and Meulder, 2003) for NER. The results are shown in Table 3 (as the space is limited, the results of NER are shown in Appendix Table 3). Note that for the Chunking and NER task, the NP Chunk and NER feature weight of CogBERT is set as 0, respectively, to ensure no data leakage problem.

We compare CogBERT with SeqVat (Chen et al., 2020), which uses virtual adversarial training to improve the model's performance and robustness on the CoNLL-2000 text chunking task. We observe that our method outperforms all baselines on the benchmark dataset, which demonstrates that CogBERT could also benefit sequence labeling tasks.

3.4 Intrinsic Evaluation

As shown in Table 4, we present an intrinsic evaluation to output the feature weight of CogBERT in different tasks, which is sampled from the Gaussian Distribution in Equation (2). Due to the space limitation, we only demonstrate feature weights of the BERT-base model in COLA, MRPC, RTE, CoNLL-2000 Chunking, and CoNLL-2003 NER tasks. The feature weight $C_f^{'} \in \mathbb{R}^i$ scaled by the data size and the feature density is calculated as:

$$
\begin{aligned}
f_d &= \frac{1}{Z}\sum_{k=1}^{z} U_k/(sum(U_k)), \\
C_s &= \frac{1}{Z}\sum_{k=1}^{z} C_k/f_d, \\
C_f^{'} &= C_s/\max(C_s),
\end{aligned}
\tag{5}
$$

where the data size of a task is $Z \in \mathbb{R}^z$, the averaged feature weight of 12 layers from CogBERT is $C \in \mathbb{R}^{r\times z}$, and the density of feature is $f_d \in \mathbb{R}^r$. C_s is the feature weight scaled by f_d, and the count of each feature in the task is $U \in \mathbb{R}^{r\times z}$ (for example, if there are 3 entity words in a training example, the count of NER feature of this example is 3).

Models	Zuco	Geco (EN)	Geco (NL)
BERT-EN	93.42(0.02)	93.68(0.14)	—
BERT-NL	—	—	91.81(0.23)
BERT-MULTI	93.74(0.05)	93.73(0.12)	91.90(0.16)
XLM-17	92.05(2.25)	91.79(1.75)	91.04(0.70)
XLM-100	93.97(0.09)	93.04(1.40)	**92.31(0.22)**
CogBERT (Random)	93.23(0.13)	93.36(0.38)	91.59(0.35)
CogBERT	**93.99(0.02)**	**93.90(0.14)**	91.97(0.09)

Table 5: Results of eye-tracking prediction. Standard deviation of 5 runs is reported in parentheses.

We find that CogBERT can assign reasonable weights to different features in various tasks. In the COLA (a linguistic acceptability judgment task), CogBERT evaluates NP Chunk (i.e., the noun phrase) as the most important feature and gives a high score to other syntax-related features. This is because the syntax structure helps judge linguistic acceptability. In the MRPC (a sentence paraphrasing task), CogBERT considers NER and emotion as the two most important features. The main reason is that if two sentences do not share the same entity and emotion, they are probably not a paraphrase. In the RTE (a text entailment task), CogBERT thinks of NP Chunk as the most important feature, and this is probably because if a sentence can be inferred from another sentence, they might have a similar phrasal structure or meaning. In CoNLL-2000 Chunking and CoNLL-2003 NER tasks, it is not surprising that CogBERT ranks NP Chunk and NER as the most crucial feature, respectively.

3.5 Eye-tracking Prediction

We argue that CogBERT can be useful not only for NLP tasks but also for language comprehension in brains. Hence, we also evaluate the effectiveness of CogBERT on eye-tracking prediction (Hollenstein et al., 2021) tasks using three eye-tracking benchmark datasets, including Zuco (Zuco 1.0 and Zuco 2.0), Geco (EN), and Geco (NL). The training details are the same with Hollenstein et al. (2021)[3]. This task tests the model's ability to learn human eye-tracking data, including first fixation duration (FFD), total reading time (TRT), number of fixa-

[3]Geco (EN) and Geco(NL) are the English and Dutch parts of Geco eye-tracking data, respectively.

Models	SST2	MRPC	RTE
CogBERT	**93.7**	**89.7**	**70.1**
w/o lower strands	93.4	89.1	68.2
w/o upper strands	92.9	88.7	67.3
w/o all strands	92.3	87.4	66.4
w/o readability	93.2	88.9	68.5
w/o layer-wise	93.1	88.5	67.9

Table 6: Results of ablation study.

tions (nFix), mean fixation duration (MFD), first pass duration (FPD), fixation proportion (FProp), number of re-fixations (NREFIX), re-read proportion (REPROP). The performance is evaluated by the mean absolute error (MAE), and we report 100–MAE as the result in this experiment.

We compare our method with BERT-EN, BERT-NL (de Vries et al., 2019), BERT-MULTI, (Wolf et al., 2020)[4] and XLM (Lample and Conneau, 2019) (XLM is a cross-lingual pre-trained language model. XLM-17 pre-trains on 17 languages and 100 for XLM-100). CogBERT is based on BERT-EN and BERT-NL, respectively. The results are shown in Table 5.

We find that CogBERT outperforms BERT-EN, BERT-NL, BERT-MULTI, and XLM-17 and achieves comparable performance with XLM-100, even though CogBERT is based on BERT, which only pre-trained on one language whereas XLM-100 pre-trained on more than 100 languages. This also shows the effectiveness of our cognitive features induced from cognitive data for understanding and explaining human gaze behavior.

3.6 Ablation Study

We conduct ablation studies over several factors related to CogBERT's performance on the downstream tasks. All results are obtained on the development sets of SST2, MRPC and QNLI, and are shown in Table 6.

We observe that replacing lower or upper strands cognitive features can decrease the model performance, and removing all strands cognitive features would further affect the model performance. We also notice that although the readability is not as important as cognitive features for our model, removing it also harms the performance. This is mainly because this factor also constrains the learning process of CogBERT from a cognitive perspective. Without layer-wise means that we integrate all features into each layer of BERT, The poor performance of without layer-wise demonstrates that incorporating features in a layer-wise manner is an effective way for cognition-guided NLP.

[4]BERT-EN, BERT-NL and BERT-MULTI, are English, Dutch and multilingual version of BERT.

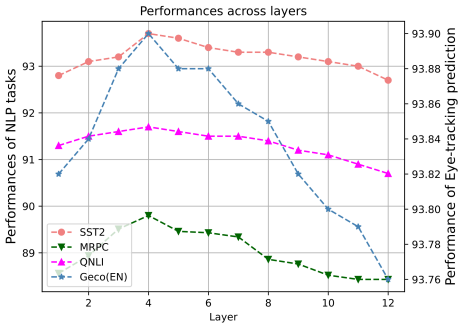

Figure 4: Performances of layer study.

3.7 Layer Study

In this section, we quantificationally discuss which layer of BERT should be the boundary for the lower and upper strands cognitive features. We conduct comparative experiments on the development sets of SST2, MRPC, QNLI and STS-B tasks, and illustrate results in Figure 4. The Y-axis is the performance for different NLP tasks. The X-axis is the number of layers. For example, if the number of layer is 6, we incorporate lower strands cognitive features into 1-6 layers of BERT and upper strands cognitive features into the rest.

We find that all tasks reach the best performance when the layer boundary is around 4, which means that BERT's lower layers are more suitable for incorporating lower strands cognitive features and upper strands cognitive features are more useful when we incorporate them in higher layers. These results can effectively guide the future research of cognition-enhanced pre-trained models. Similarly, previous research (Tenney et al., 2019) also finds that the BERT behaves like a pipeline manner where the lower layer processes basic information and upper layer processes the sentence based on the information of previous layers, in which its argument is similar to ours and our research further validates their research and further proved this argument from the perspective of model implementation level.

3.8 Attention Visualization

To qualitatively analyze the effectiveness of our method, we visualize the attention case of CogBERT and compare it with BERT and humans. We select cases from SST2, NER and MRPC tasks. To compare with human cognition, given a specific

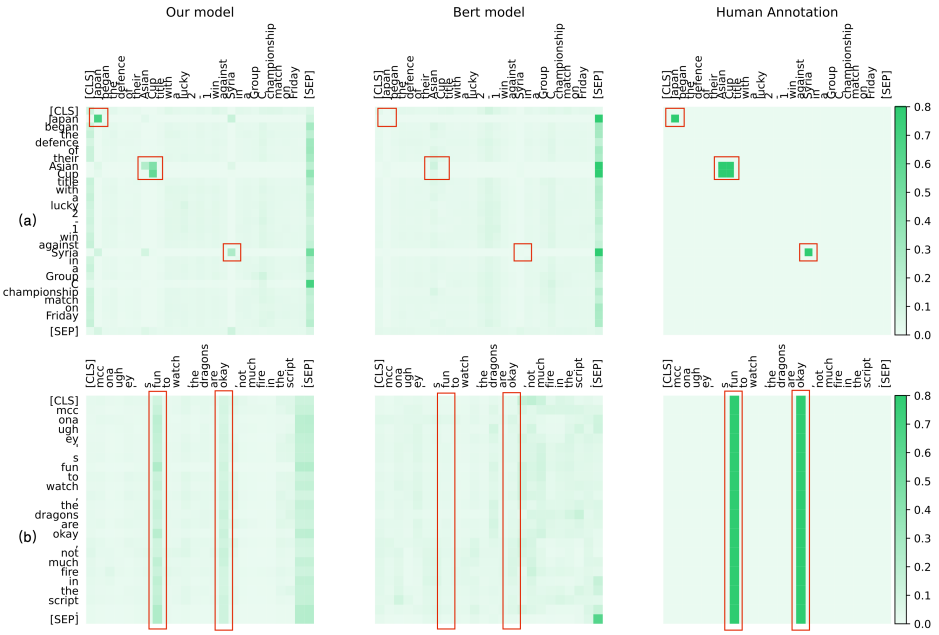

Figure 5: Attention cases selected from the SST2 and CoNLL-2003 NER task, respectively.

NLP task, we ask four annotators to highlight their attention words when reading the sentences. For BERT and CogBERT, we select attention scores from higher layers of pre-trained models, which can capture task-specific features (Merchant et al., 2020). The attention visualizations of SST2 and NER are illustrated in Figure 5 (MRPC is shown in Appendix Figure 1).

Figure 5 (a) presents the attention visualization for the CoNLL-2003 NER task, illustrating that CogBERT pays more attention to NER words 'Asian Cup', 'Japan' and 'Syria' like humans, whereas BERT gives little attention to these words. Figure 5 (b) illustrates the attention visualization for the SST2 task, showing that CogBERT captures the critical emotion words 'fun' and 'okay', which are also important for the human judgments. In contrast, BERT fails to focus on these words. These experimental results indicate that although pre-trained models have achieved promising improvements in numerous NLP tasks, they are still far from the level of human intelligence. Cognition-guided pre-trained models can provide an effective way to approach human cognition, by learning the attention mechanism in human reading.

4 Related Work

Prior neuroscience studies have demonstrated that cognitive data is associated with language comprehension activity in human brains (Just and Carpenter, 1980; Brooks and Meltzoff, 2005), showing longer duration links to a greater processing load in different language units (words, clauses, texts). These studies established the theoretical grounding for cognition-guided NLP.

In cognitively motivated NLP, researchers investigate the impact of cognitive language processing signals on NLP tasks, especially focusing on improving neural networks by utilizing cognitive data. Early researches mainly used LSTM or CNN to incorporate cognitive signals (Barrett et al., 2018; González and Søgaard, 2018; Hollenstein and Zhang, 2019; Sood et al., 2020; Ren and Xiong, 2021; Takmaz et al., 2020).

As pre-trained models have shown their great power on various NLP tasks, a line of work focuses on exploring cognitive-data-enhanced pre-trained models (Hollenstein et al., 2019, 2021; McGuire and Tomuro, 2021), mainly by fine-tuning pre-trained models on cognitive data.

However, these methods cannot give a fine-grained analysis of how cognitive data contributes to different NLP tasks. In contrast, CogBERT is inspired by the theory in psycholinguistics and encodes cognitive features induced from eye-tracking data into pre-trained models in a layer-wise manner with carefully designed architecture, enabling us to perform a fine-grained analysis of how cognitive data contributes to different NLP tasks and further improved the performance of pre-trained models comparing to previous simply fine-tuned ones.

5 Conclusion

We present CogBERT, a framework that can effectively incorporate cognitive signals into pre-trained models. Experimental results show that CogBERT achieves new SOTA results on three NLP benchmark datasets. Analyses suggest that CogBERT can adaptively learn task-specific cognitive feature weights to give fine-grained explanations of how cognitive data works on NLP tasks. This work provides a new paradigm in learning cognition-enhanced pre-trained models, and extensive elaborated experiments can guide future researches.

References

Jiangang Bai, Yujing Wang, Yiren Chen, Yaming Yang, Jing Bai, Jing Yu, and Yunhai Tong. 2021. Syntax-bert: Improving pre-trained transformers with syntax trees.

Maria Barrett, Joachim Bingel, Nora Hollenstein, Marek Rei, and Anders Søgaard. 2018. Sequence classification with human attention. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 302–312, Brussels, Belgium. Association for Computational Linguistics.

Rechele Brooks and Andrew Meltzoff. 2005. The development of gaze following and its relation to language. *Developmental science*, 8:535–43.

Luoxin Chen, Weitong Ruan, Xinyue Liu, and Jianhua Lu. 2020. SeqVAT: Virtual adversarial training for semi-supervised sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8801–8811, Online. Association for Computational Linguistics.

Uschi Cop, Nicolas Dirix, Denis Drieghe, and Wouter Duyck. 2016. Presenting geco: An eyetracking corpus of monolingual and bilingual sentence reading. *Behavior Research Methods*, 49.

Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. 2019. Bertje: A dutch bert model.

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

Ana Valeria González and Anders Søgaard. 2018. Learning to predict readability using eye-movement data from natives and learners. In *AAAI*.

Mickel Hoang, Oskar Alija Bihorac, and Jacobo Rouces. 2019. Aspect-based sentiment analysis using BERT. In *Proceedings of the 22nd Nordic Conference on Computational Linguistics*, pages 187–196, Turku, Finland. Linköping University Electronic Press.

Nora Hollenstein, Antonio de la Torre, Nicolas Langer, and Ce Zhang. 2019. Cognival: A framework for cognitive word embedding evaluation.

Nora Hollenstein, Federico Pirovano, Ce Zhang, Lena Jäger, and Lisa Beinborn. 2021. Multilingual language models predict human reading behavior.

Nora Hollenstein, Jonathan Rotsztejn, Marius Troendle, Andreas Pedroni, Ce Zhang, and Nicolas Langer. 2018. Zuco, a simultaneous eeg and eye-tracking resource for natural sentence reading. *Scientific Data*, 5:180291.

Nora Hollenstein, Marius Troendle, Ce Zhang, and Nicolas Langer. 2020. Zuco 2.0: A dataset of physiological recordings during natural reading and annotation.

Nora Hollenstein and Ce Zhang. 2019. Entity recognition at first sight: Improving ner with eye movement information.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Marcel Adam Just and Patricia A. Carpenter. 1980. A theory of reading: from eye fixations to comprehension. *Psychological review*, 87 4:329–54.

J. Peter Kincaid, Robert P. Fishburne, Richard Lawrence Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel.

Reinhold Kliegl, Ellen Grabner, Martin Rolfs, and Ralf Engbert. 2004. Length, frequency, and predictability effects of words on eye movements in reading. *European Journal of Cognitive Psychology - EUR J COGN PSYCHOL*, 16:262–284.

Victor Kuperman, Michael Dambacher, Antje Nuthmann, and Reinhold Kliegl. 2010. The effect of word position on eye-movements in sentence and paragraph reading. *Quarterly journal of experimental psychology (2006)*, 63:1838–57.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

Zhongli Li, Qingyu Zhou, Chao Li, Ke Xu, and Yunbo Cao. 2021. Improving bert with syntax-aware local attention.

Zhongyang Li, Xiao Ding, and Ting Liu. 2019. Story ending prediction by transferable bert.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Erik McGuire and Noriko Tomuro. 2021. Relation classification with cognitive attention supervision. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 222–232, Online. Association for Computational Linguistics.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to bert embeddings during fine-tuning?

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. volume 14, pages 1532–1543.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Yuqi Ren and Deyi Xiong. 2021. Cogalign: Learning to align textual neural representations to cognitive language processing signals.

Erik Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. *Proc of CoNLL-2000 and LLL-2000*.

Erik Sang and Fien Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *Proceeding of the Computational Natural Language Learning (CoNLL)*.

H.S. Scarborough, F. Fletcher-Campbell, Janet Soler, and G. Reid. 2009. Connecting early language and literacy to later reading (dis)abilities: Evidence, theory, and practice. *Approaching difficulties in literacy development: assessment, pedagogy, and programmes*, pages 23–39.

Dan Schwartz, Mariya Toneva, and Leila Wehbe. 2019. Inducing brain-relevant bias in natural language processing models.

Timothy Slattery and Mark Yates. 2017. Word skipping: Effects of word length, predictability, spelling and reading skill. *The Quarterly Journal of Experimental Psychology*, 71:1–30.

Ekta Sood, Simon Tannert, Philipp Mueller, and Andreas Bulling. 2020. Improving natural language processing tasks with human gaze-guided neural attention.

Ece Takmaz, Sandro Pezzelle, Lisa Beinborn, and Raquel Fernández. 2020. Generating Image Descriptions via Sequential Cross-Modal Alignment Guided by Human Gaze. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4664–4677, Online. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Mariya Toneva and Leila Wehbe. 2019. Interpreting and improving natural-language processing (in machines) with natural language-processing (in the brain).

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

A Appendix

A.1 Training Details

A.1.1 Bi-LSTM Model Training

The eye-tracking data to train the Bi-LSTM model contains 1637 sentences and 76,937 words. The embedding size of Glove Vector (Pennington et al., 2014) is 300, which is accessible on the page.[5] We used the embedding trained on Common Crawl corpus, and the embedding is not trained along with the Bi-LSTM model. The hidden size of Bi-LSTM is 256, the dropout ratio in the feed-forward network is 0.15. We train the Bi-LSTM model with 40 epochs with learning rate 1e-3, and we use the AdamW optimizer(Loshchilov and Hutter, 2019). We evaluate the $Loss_G$ on the test set of nFix and select the model with the best performance.

A.1.2 CogBERT Model Training

For Transformers, we set layer number as 12 to keep the layer setting the same with the different base versions of pre-train models, the hidden dimension of intermediate layers as 512, dropout

[5] https://github.com/stanfordnlp/GloVe

ratio as 0.15, and the dimension of the fully connected layer before Softmax activation as 2000. The learning rate is initialized as 5e-4 for the AdamW optimizer, and we use the linear learning rate schedule with the warm-up ratio of 0.1.

For BERT and RoBERTa, we use the pre-trained version of different models released by HuggingFace.[6]For GLUE Benchmark. We use AdamW optimizer with learning rate from [1e-5, 3e-5, 5e-5] on GLUE benchmark. The training epochs are from [3, 5, 10]. We also use the linear learning rate schedule with a warm-up ratio of 0.1. The batch size is 32 for all tasks. The kernel size for both 1DCNN and 2DCNN is 3. The 2DCNN module does not have bias, and the padding strategy is the "same".

For CoNLL-2000 chunk and CoNLL-2003 NER tasks, we use AdamW optimizer with learning rate from [1e-6, 3e-6, 5e-6]. The training epochs are from [40, 60]. We also use the linear learning rate schedule with a warm-up ratio of 0.1. The batch size is 32 with a gradient accumulation step of 4 to avoid memory issues.

For the Eye-tracking Prediction task, we follow Hollenstein et al. (2021) and replace the BERT model with our model. The training setting is precisely the same as the original task setting released on the page.[7] The results are AdamW optimizer with a learning rate of 5e-5 and a weight decay of 0.01. We employ a linear learning rate decay schedule over the total number of training steps. We clip all gradients exceeding the maximal value of 1. We train the models for 100 epochs, with early stopping after seven epochs without improving the validation accuracy.

A.2 Eye-tracking Feature Validation

For word-level features, the results are computed using the percentage of the word that carries a specific feature with a higher nFix value than sentence average, which means the feature is a strong indicator for this word that carries this feature to be fixated or not.

For semantic features, the strategy is different. For noun phrases, we take the nFix of the first word and found that the first word of nFix always has a below-average nFix value, which shows that a below-average first word of a phrase is an indicator to show that this is a noun phrase. We take the nFix average of the emotion phrase and compare it with the sentence average nFix value. For dependency relation, we check the equality relation between the nFix of two words, and the percentage is computed using that the ratio of a head word has a nFix that is greater or less than the tail word. We checked 46 features that cover a large portion of existing syntax features, and the results are in Table 7.

Additionally, some dependency relations are very uncommon, which does not hold a statistical count above one hundred in those eye-tracking corpora. We do not use uncommon relations with less than 100 counts in the corpus, which we label these relations as sparse in the Table 7. In this research, we do not consider any relationship that is sparse in the model. We hope this table is helpful for future research in this area. Since word length and position are already proven to be directly connected to eye-tracking data in psycholinguistics (Slattery and Yates, 2017;Kliegl et al., 2004;Kuperman et al., 2010), therefore we do not further validate these two features again in our research.

A.3 CoNLL-2003 NER Task

We are not able to reproduce the BERT results in CoNLL-2003 NER tasks, which is also discussed in different GitHub issues[8][9][10]. The official GitHub page of BERT says they used a complicated pre-processing script to process the entity word before training using BERT. However, they are not planning to release the pre-processing script.

A.4 MRPC Attention Case

Besides SST2 and CoNLL-2003 NER task, we also present another attention case in MRPC. In this example. Sentence 1 is "*Sanitation is poor.. there could be typhoid and cholera*" and sentence 2 is "*Sanitation is poor drinking water is generally left behind..there could be typhoid and cholera*". These two sentences are a paraphrase, which states poor sanitation might cause typhoid and cholera.

From the cognition of humans, human labels "*Sanitation is poor*," "*typhoid*" and "*cholera*" as the keywords to label these two sentences as a paraphrase.

The attention of CogBERT, BERT, and humans is in Figure 6. Since this is a classification task

[6] https://huggingface.co/transformers/

[7] https://github.com/DS3Lab/multilingual-gaze

[8] https://github.com/dmlc/gluon-nlp/issues/593

[9] https://github.com/kyzhouhzau/BERT-NER/issues/2

[10] https://github.com/google-research/bert/issues/223

Feature	Zuco 2.0	Zuco 1.0	Geco	Avg	Sparse	Trands
Word Length	✓	✓	✓	✓	✓	Lower
Word Position	✓	✓	✓	✓	✓	Lower
Punctuation	×	×	×	×	×	Lower
Marker	29.60%	29.41%	31.83%	30.28%	True	Lower
ContentWord	96.0%	98.20%	99.50%	97.90%	False	Lower
NER	67.70%	68.90%	83.10%	73.20%	False	Lower
Determiner	7.60%	10.80%	14.06%	10.82%	False	Lower
Negation	33.30%	43.62%	47.00%	41.31%	False	Lower
Emotion	75.00%	66.67%	88.89%	93.85%	False	Upper
NPChunk	75.00%	89.30%	85.60%	83.30%	False	Upper
Parent-Child	54.80%	55.20%	62.00%	57.33%	False	Upper
Token-Head	46.00%	47.90%	43.82%	46.00%	False	Upper
Word and subtree	56.67%	54.81%	64.49%	58.66%	False	Upper
Adjective Modifier	63.79%	62.85%	57.40%	61.35%	False	Upper
Noun Subject	35.88%	49.17%	32.47%	39.17%	False	Upper
Compound	60.79%	51.30%	35.58%	49.22%	False	Upper
Adjective Complement	81.30%	60.30%	85.76%	75.79%	False	Upper
Adjective Clause	52.94%	41.00%	54.50%	49.48%	True	Upper
Adverbial Clause Modifier	51.85%	59.45%	58.70%	56.67%	False	Upper
Adverbial Modifier	52.03%	46.70%	53.05%	50.59%	False	Upper
Agent	0.00%	5.71%	5.33%	3.68%	True	Upper
Appositional Modifier	40.62%	43.50%	58.30%	47.47%	True	Upper
Auxiliary	12.87%	21.95%	30.18%	21.67%	False	Upper
Passive Auxiliary	24.72%	14.38%	10.88%	16.66%	False	Upper
Coordinating Conjunction	5.71%	11.34%	17.46%	11.50%	False	Upper
Clausal Complement	53.13%	48.45%	48.52%	50.03%	False	Upper
Conjunct	51.34%	48.59%	53.05%	50.99%	True	Upper
Clausal Subject	0.00%	100%	83.33%	61.00%	True	Upper
Dative	33.33%	36.36%	25.19%	31.63%	False	Upper
Direct Object	38.55%	43.58%	37.32%	39.82%	True	Upper
Expletive	0.00%	76.47%	78.70%	52.00%	True	Upper
Interjection	—	0.00%	44.23%	22.00%	True	Upper
Meta Modifier	0.00%	0.00%	0.00%	0.00%	True	Upper
Modifier of Nominal	70.00%	48.83%	44.44%	54.00%	False	Upper
Noun Phrase as Adverbial Modifier	60.29%	52.90%	59.09%	57.43%	False	Upper
Passive Nominal Subject	35.53%	32.43%	31.29%	33.08%	False	Upper
Numeric Modifier	41.13%	34.24%	31.29%	33.08%	False	Upper
Object Predicate	60.00%	54.54%	59.36%	58.00%	True	Upper
Parataxis	50.00%	12.50%	63.92%	42.00%	True	Upper
Complement of Preposition	66.67%	85.71%	88.15%	80.18%	False	Upper
Object of Preposition	91.21%	91.50%	84.43%	89.05%	False	Upper
Possession Modifier	29.00%	34.16%	26.56%	30.00%	False	Upper
Pre-correlative Conjunction	0.00%	47.05%	64.71%	37.00%	True	Upper
Predet	0.00%	14.29%	36.90%	17.00%	True	Upper
Prepositional	8.27%	12.44%	16.77%	12.00%	True	Upper
Particle	0.00%	32.25%	20.85%	18.00%	True	Upper
Modifier of Quantifier	70.00%	50.00%	20.05%	47.00%	True	Upper
Relative Clause Modifier	34.00%	40.00%	49.79%	41.00%	True	Upper
Open Clausal Complement	41.17%	40.43%	46.65%	42.75%	True	Upper

Table 7: Validation Table of Features nFix Data. — means this relation does not exist in this corpus based on the annotation result of Spacy. ×means this relation cannot be checked. ✓means this relation has been verified by previous research. Sparse means the number of this relation in any corpus is less than 100.

between two sentences, a [SEP] token will be inserted between them. In an ideal attention situation, for the paraphrase classification task, sentence 1 should attend to the same words of sentence 2, and sentence 2 should attend to the same words of sentence 1, respectively.

From the Figure 6, we can see that CogBERT can align more accurately with human annotation. CogBERT can give more attention to "*Sanitation is poor*" from sentence 1 to sentence 2 and from sentence 2 to sentence 1. In contrast, the BERT model gives very little attention to this phrase. Additionally, CogBERT is also able to focus on the "*typhoid*" word, which is tokenized into several subwords. However, the BERT model is not able to capture this in both sentences. Moreover, for the word "*cholera*," even both CogBERT and BERT can focus on it, CogBERT can give more attention compared to the BERT model.

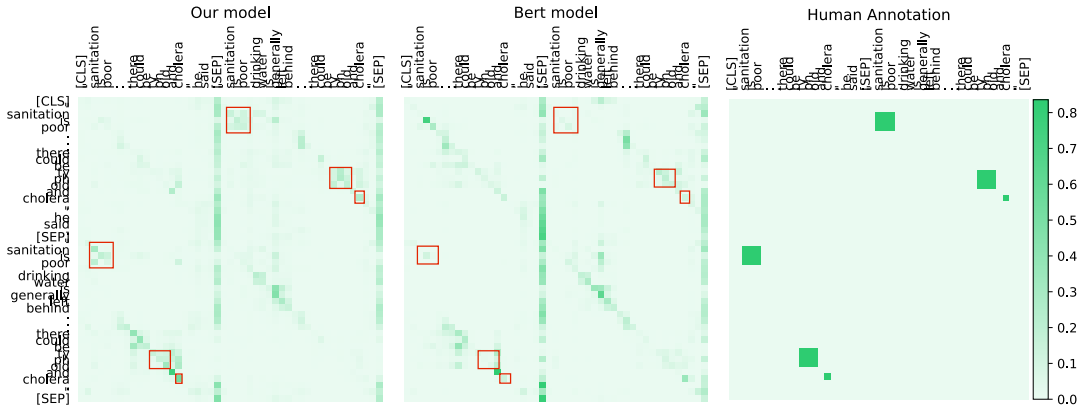

Figure 6: MRPC Attention Case

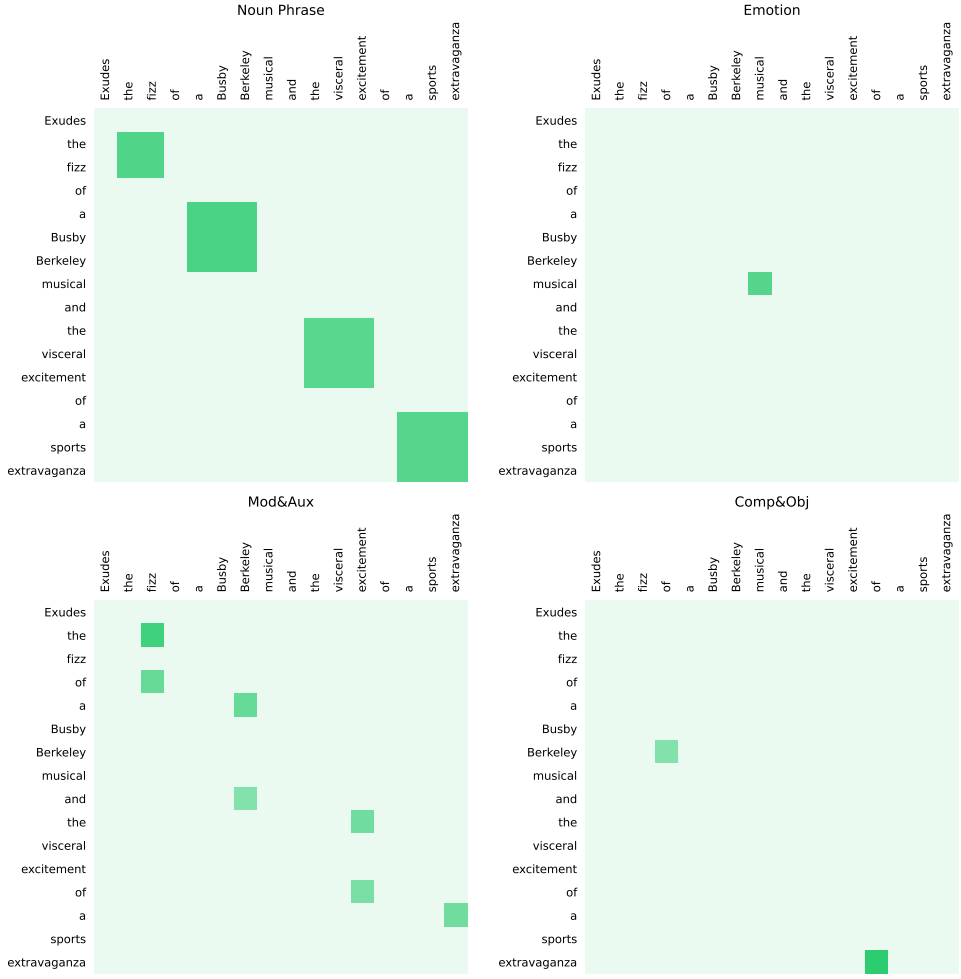

Figure 7: Upper strands feature matrices.

A.5 Model Efficiency

Since CogBERT works in a two-stage setting and uses Spacy to annotate the sentence, it is necessary to report the processing speed to show that processing efficiency is not a significant concern in our work.

A.5.1 Processing Efficiency

First, we report the training time to train the Bi-LSTM model. In a 2GHz four-core Intel Core i5, training the Bi-LSTM model requires 30 minutes in 40 epochs.

The most time-consuming part is the Spacy annotation processing. In the same chip setting, take

Models	P (%)	R (%)	F1 (%)
LISA-Base	90.7	92.2	91.4
SGNET-Base	90.9	92.6	91.7
BERT-Base (Devlin et al., 2019)	—	—	**92.4**
BERT-Base (Li et al., 2021)	91.0	92.3	91.6
CogBERT-Base (Random)	90.3	91.2	90.7
CogBERT-Base	**91.4**↑	**92.6**↑	91.9↑
LISA-Large	91.3	92.6	92.0
SGNET-Large	91.5	92.8	92.1
BERT-Large (Devlin et al., 2019)	—	—	**92.8**
BERT-Large (Li et al., 2021)	91.7	93.1	92.4
CogBERT-Large (Random)	91.1	91.4	91.2
CogBERT-Large	**92.0**↑	**93.2**↑	92.6 ↑

Table 8: Results of CoNLL-2003 NER. CogBERT is based on BERT-Cased. ↑ means that we outperform previous implementation of BERT (Li et al., 2021).

SST2 tasks for example; the training part of the SST2 dataset contains 67,349 examples. In this amount of data, producing feature vectors/matrices with Spacy's annotation for lower and upper strands will take about 15 minutes in different runs. For much larger datasets like QNLI that contains more than 100,000 training examples, running the generation process will take about 40 minutes.

Since most of the data in our experiments are less than 10,000 examples, we do not think the feature vector/matrix generation process is a problem in our model. The generation processing of most tasks can be done in 10 minutes.

A.5.2 Storage Efficiency

Another point to mention is the data size of generated feature vector and matrix. We used a sparse matrix to store the generated feature vector/matrix, which enables us to store them efficiently and recover them in only several lines of code.

Take SST2 for example, the size of generated feature vector/matrix of the whole data is 158.3 Mb, and 708.5 Mb for QNLI, roughly ten times the size of the text version. Since text size is considerably small in different tasks, our generated vector/matrix does not use too much disk storage. Therefore disk storage is not a concern in most tasks.

A.5.3 Training Efficiency

For the training speed of the second stage, since we did not put too many parameters in our model, which are just several linear transformations and two CNN networks. For example, in the MRPC task, the total parameter size of the BERT model is 109M, and the total parameter size of our model is 116M.

Additionally, the CNN network is efficient since the architecture of CNN is suitable for parallel computation. Thus, the training speed in downstream tasks is close to the training speed in different base pre-train models implementation.

Models	nFix Loss	Feature Loss
Bi-LSTM	0.092	—
Bi-LSTM (with feature)	**0.051**	**0.006**

Table 9: Results of nFix prediction. We sample 20% nFix data as test set. The evaluation criteria is the MSE loss.

A.6 nFix Prediction

First, we report the results of the nFix prediction in Table 9. The base model is Bi-LSTM with four layers only trained with nFix prediction, whereas our model will first predict feature score and further predict the nFix following our description.

From the results, we could tell that by training with loss L_T. Our model gives a lower loss, which means the proposed training method can give a more accurate nFix estimation by combining different feature scores. This proves that these features help predict the nFix value, and the model can assign adaptive weight to different features and combine them to better estimate nFix.

A.7 The Effect of Spacy Tool

From the noun phrase in Figure 8, we could see the annotation is not perfect, like the *a Busby Berkeley*, this phrase is only a part of the whole noun phrase.

Reasonably, the performance of the Spacy tool certainly affects our model's performance. However, the annotation of Spacy is correct in most situations. Details about the performance of the Spacy tool are on the official website.[11].In this research, we use the medium size Spacy annotation model in all tasks. For the Eye-tracking Prediction task of Geco(NL), we also use the medium-sized Netherlands model released by Spacy.

A.8 Sample Input/Output of Bi-LSTM Model

We present a sample Bi-LSTM prediction case in Figure 8, the generated feature vectors for lower strands are in Figure 9, and the generated feature matrices for upper strands are in Figure 7.

Figure 9 presents the generated feature vectors of lower strands, including Word Length, Word Position, NER, and Content Words. These features correspond to a single word, which means these features are word-level rather than phrase or sentence level features. Therefore the output of the

[11] https://spacy.io/

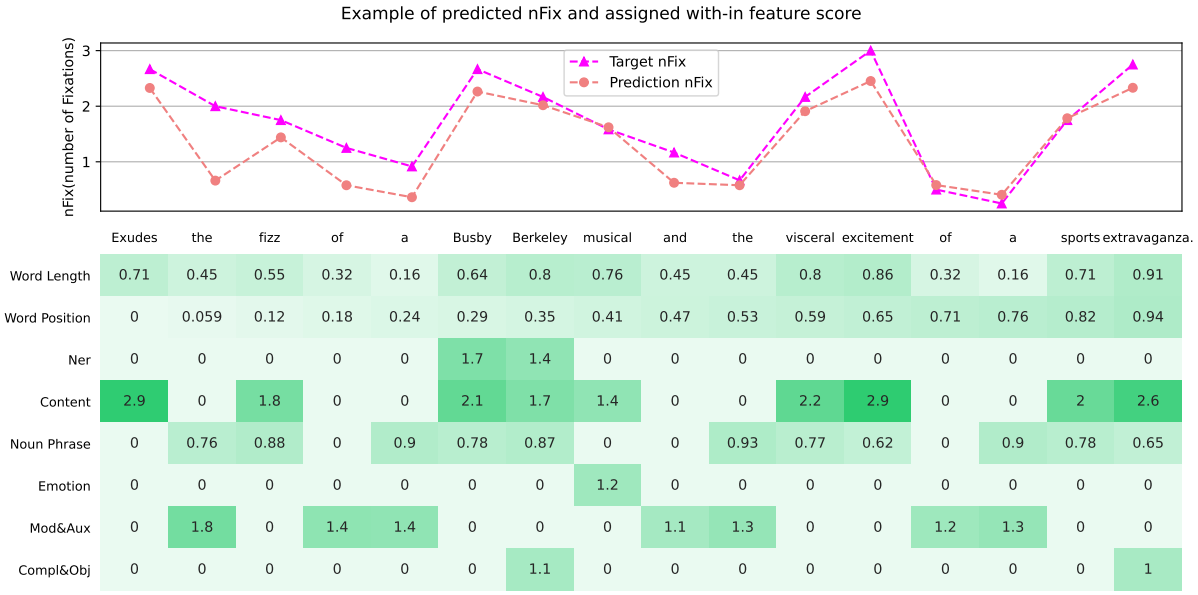

Figure 8: Bi-LSTM Feature Prediction. The number inside each cell of the prediction map are weighted scores computed by the Bi-LSTM model. The top part is the nFix prediction given by our model, the brown line means the prediction, and the pink line means the target nFix measured by human gaze.

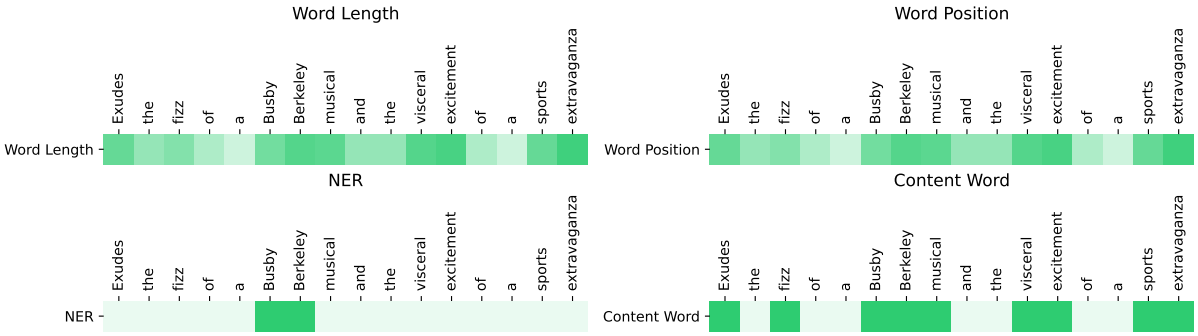

Figure 9: Lower strands feature vectors.

Bi-LSTM model will be directly used as feature vectors for different lower strands features.

Figure 7 presents the generated feature matrices of upper strands containing Noun Phrase, Emotion, Mod&Aux, and Comp&Obj. These features might correspond to several words or a phrase, which means they are phrase-level features compared to lower strands features.

For noun phrase, we take the average weight from the prediction of the Bi-LSTM model of words in the same noun phrase, then we assign the average weight to the noun phrase feature matrix, if the phrase length is N, then we have a noun phrase feature matrix $N \times N$ .In this example, we take the noun phrase "*the fizz*" for example, the Bi-LSTM predicts 0.76 and 0.88 for those two words in the noun phrase dimension. The average weight is 0.82 for this phrase, and then we fill a 2×2 matrix with this value in the whole feature matrix. This process is the same for both noun phrase feature and emotion feature.

For dependency relations, if word w_i is the tail word of w_j of the dependency relation we use in this paper, then we assign the prediction of weight for w_i in this relation to E_{ij}, where E is the feature matrix of this feature. Take first "*the*" word of this sentence in the Mod&Aux dimension, this word is the determiner of the word "*fizz*," and "*the*" is the tail word of the head word "*fizz*." The prediction value from Bi-LSTM for word "*the*" in Mod&Aux dimension is 1.8. Following our generation rules, we assign 1.8 from word "*the*" to word "*fizz*" in the corresponding position of the feature matrix of Mod&Aux. In this case, we assign 1.8 to E_{23}. This process is the same for both Mod&Aux and Comp&Obj features.

A.9 Attention Annotation

To compare the attention produce by CogBERT, BERT and human, we select several cases from MRPC, SST2 and NER cases and asked four annotators, which are all students majoring in Computer Science and have background knowledge about NLP, to highlight the words that they think are important for this task. We then collect the average importance given by the annotators which represents as the importance score for words in that sentence for different tasks.