Do Flawless Transcripts Matter? Evaluating Different Input Levels for PHQ-8 Depression Prediction

Haike Yu, Sun Bin Mun, Edbert Wang, Chengyuan Yao
{hyu606, smun6, ewang437, cyao81}@gatech.edu
Georgia Institute of Technology

Abstract

Predicting PHQ-8 depression severity from patient–therapist transcripts is an important goal in clinical NLP. Yet it remains unclear whether the granularity of input text—word, sentence, or dialogue—meaningfully affects model performance. To investigate this question, we conduct a systematic comparison across four commonly used text models—Transformer, TextCNN, Attention-RNN, and LSTM—evaluated under matched training conditions. All models are first tuned on dialogue-level inputs and then applied to sentence- and word-level variants of the same transcripts, providing a controlled assessment of how textual granularity contributes to PHQ-8 prediction accuracy.

1 Introduction

Predicting PHQ-8 depression severity from patient–therapist transcripts has become an important direction in clinical NLP. However, despite the growing interest in automated mental-health assessment, it remains unclear how the *granularity of textual input* affects a model's ability to capture depressive signals. Prior work often focuses on a single representation level—but does not systematically evaluate how these levels interact with different neural architectures.

To address this gap, we conduct the first systematic comparison of three textual resolutions: **word-level** (randomly sampled individual tokens), **sentence-level** (randomly sampled full sentences), and **dialogue-level** (multi-sentence conversational segments). These levels represent increasingly coherent linguistic structure, from isolated lexical cues to full conversational context.

Our goal is to determine whether finer granularity or broader conversational structure contributes more to PHQ-8 severity prediction, and whether certain architectures are more sensitive to these differences.

Model	Word Input	Sentence Input	Dialogue Input
Transformer	Word+Trans	Sent+Trans	Dial+Trans
TextCNN	Word+CNN	Sent+CNN	Dial+CNN
Attention-RNN	Word+Att	Sent+Att	Dial+Att
LSTM	Word+LSTM	Sent+LSTM	Dial+LSTM

Table 1: All 12 model–input configurations evaluated in this study.

2 Methods

2.1 Dataset

We conduct our study using the DAIC-WOZ dataset [1], a widely adopted benchmark for automated depression assessment. The corpus consists of semi-structured clinical interviews between participants and a virtual agent, accompanied by PHQ-8 depression scores that serve as the ground-truth regression targets. DAIC-WOZ provides audio recordings, automatic transcripts, and session-level metadata. The PHQ-8 is an eight-item clinical instrument measuring depressive symptoms over the past two weeks. Each item is scored from 0 to 3, yielding a total severity score from 0 to 24. The questionnaire assesses core symptom domains including anhedonia, depressed mood, sleep disturbance, fatigue, appetite or weight changes, feelings of worthlessness, concentration difficulties, and psychomotor agitation or retardation. In this study, we use only the interview transcripts as model input and the PHQ-8 scores as the corresponding prediction targets.

2.2 Preprocessing

Given our objective of examining how different textual granularities influence PHQ-8 prediction, our preprocessing pipeline is designed to convert the raw DAIC–WOZ interviews into three parallel forms of input: word-level, sentences-level, and dialogue-level. Beginning with the original transcripts and metadata, we extract patient-only language and restructure it into these three levels of linguistic coherence. This transformation enables a controlled comparison across models as they operate on increasingly rich contextual units. The complete preprocessing workflow consists of four stages, described below.

2.2.1 Data Acquisition and Parsing

We begin by downloading the official DAIC–WOZ ZIP archives and extracting all interview files. For each participant, we load the automatic transcription and retrieve the corresponding PHQ-8 score vector using the participant ID from the metadata CSV file. Only sessions with complete transcripts and PHQ-8 labels are retained.

2.2.2 Speaker Filtering and Utterance Extraction

Each transcript contains alternating turns between the participant and the virtual interviewer. To isolate linguistic cues associated with depressive severity, we remove all interviewer prompts and retain only the participant's spoken utterances. The resulting patient-only transcript is used for all downstream input constructions.

2.2.3 Construction of Multi-Resolution Inputs

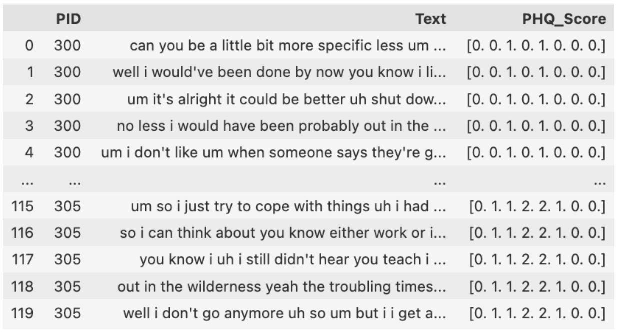
To examine the effect of textual granularity, we generate three input formats from each cleaned transcript: **Word-level:** We randomly sample 512 individual tokens and pad the sequence as necessary. **Sentence-level:** We randomly sample 512 full sentences. **Dialogue-level:** Starting from a random utterance, we extract a contiguous segment and extend it until reaching approximately 512 tokens, padding if needed. These three representations capture increasing degrees of linguistic structure, ranging from isolated words to full conversational segments.

2.2.4 Balanced Sampling Strategy

PHQ-8 scores in DAIC–WOZ exhibit substantial imbalance across severity levels. To address this, we employ an inverse-frequency weighted sampler during dataloader construction. Each interview is assigned a sampling weight proportional to the inverse of its label frequency, ensuring that rare PHQ-8 scores are represented proportionally during training.

2.2.5 Output Formatting

Finally, we store the processed dataset in CSV format, containing for each participant: the session ID, the cleaned patient-only text, and the PHQ-8 label vector. This CSV serves as the unified input for all subsequent model-specific dataloaders (see Figure 1).



	PID	Text	PHQ_Score
0	300	can you be a little bit more specific less um ...	[0. 0. 1. 0. 1. 0. 0. 0.]
1	300	well i would've been done by now you know i li...	[0. 0. 1. 0. 1. 0. 0. 0.]
2	300	um it's alright it could be better uh shut dow...	[0. 0. 1. 0. 1. 0. 0. 0.]
3	300	no less i would have been probably out in the ...	[0. 0. 1. 0. 1. 0. 0. 0.]
4	300	um i don't like um when someone says they're g...	[0. 0. 1. 0. 1. 0. 0. 0.]
...	...	...	...
115	305	um so i just try to cope with things uh i had ...	[0. 1. 1. 2. 2. 1. 0. 0.]
116	305	so i can think about you know either work or i...	[0. 1. 1. 2. 2. 1. 0. 0.]
117	305	you know i uh i still didn't hear you teach i ...	[0. 1. 1. 2. 2. 1. 0. 0.]
118	305	out in the wilderness yeah the troubling times...	[0. 1. 1. 2. 2. 1. 0. 0.]
119	305	well i don't go anymore uh so um but i i get a...	[0. 1. 1. 2. 2. 1. 0. 0.]

Figure 1: Processed dataset example showing participant ID, cleaned patient-only transcript, and PHQ-8 label vector.

2.3 Modeling Overview

After preprocessing, our dataset is transformed into three parallel input formats: a *word-level* version containing randomly sampled individual tokens, a *sentence-level* version composed of randomly sampled full sentences, and a *dialogue-level* version constructed from contiguous multi-sentence conversational segments. These three representations allow us to systematically examine how reducing or expanding contextual structure affects downstream depression prediction. We pair each input format with four neural architectures commonly used in clinical NLP—Transformer, TextCNN, Attention-RNN, and LSTM—yielding a total of twelve model–input configurations (Table 1). Our training strategy follows a controlled two-phase design. First, for each architecture, we tune the model exclusively on the *dialogue-level* inputs, which provide the richest contextual information. This ensures that all models reach stable convergence and reasonable predictive performance under maximal context. Once tuned, we freeze the hyperparameters and apply the same model to the *sentence-level* and *word-level* inputs without further adjustment. This setup isolates the effect of input granularity and enables direct comparison of how each model degrades—or remains robust—when contextual information is progressively removed.

3 Experiments

3.1 Transformer

3.1.1 Architecture Summary

We adopt the CORAL ordinal-regression framework because PHQ-8 item scores are ordered categories (0–3) rather than continuous values. Standard regression ignores this ordinal structure, whereas treating the four levels as unrelated classes loses the graded nature of errors (e.g., predicting 2 instead of 3 is less severe than predicting 0 instead of 3). CORAL models these ordered thresholds through a sequence of cumulative binary decisions, making it well matched to the structure of PHQ-8.

Under this formulation, the model predicts eight questionnaire items, each represented through three ordinal boundaries, yielding 24 logits reshaped into an (8,3) tensor. Each boundary corresponds to the decision of whether the symptom severity exceeds level 0, 1, or 2.

To generate these logits, our Transformer-based model uses only the encoder stack: a token embedding layer with layer normalization, positional encodings, a stack of multi-head self-attention encoder blocks, an attention-pooling module that aggregates the sequence into a fixed-length vector, and a final linear layer that outputs the 24 CORAL logits. (see the layer breakdown below)

For completeness, we also implemented alternative Transformer heads (e.g., regression and multi-class classification), but these formulations either ignored the ordinal nature of PHQ-8 or yielded inferior stability. CORAL therefore represents the most principled and effective choice for our study; full implementations of all variants are provided in our GitHub repository. **(1) Embedding Layer.** The input sequence is mapped into embeddings and normalized:

$$x_0 = \mathrm{LayerNorm}(\mathrm{Emb}(x)).$$

(2) Positional Encoding. We add positional encodings:

$$x_1 = \mathrm{PosEnc}(x_0).$$

(3) Transformer Encoder Stack. A stack of multi-head self-attention blocks processes the sequence:

$$x_2 = \mathrm{FinalLayerNorm}(\mathrm{TransformerEncoder}(x_1)).$$

(4) Attention Pooling. Token embeddings are aggregated using attention pooling:

$$h = \mathrm{LayerNorm}(\mathrm{AttPool}(x_2)).$$

(5) Output Projection. A fully connected layer predicts 24 ordered logits:

$$z = \mathrm{FC}(h) \in \mathbb{R}^{24}.$$

(6) Ordinal Reshaping (CORAL). Logits are reshaped into the PHQ-8 ordinal format:

$$z = \mathrm{reshape}(z; 8, 3) \in \mathbb{R}^{8\times 3}.$$

3.1.2 Training Configuration

Table 2 summarizes the key hyperparameters used to train the Transformer model under all three input granularities.

Setting	Value
Max sequence length	512
Batch size	64
Optimizer	AdamW
Learning rate	2×10^{-5}
Weight decay	3×10^{-5}
Gradient clipping	1.0
Epochs	20
Hidden dimension	128
Number of encoder layers	3
Number of attention heads	4
Dropout	0.2

Table 2: Training configuration for the Transformer-based PHQ-8 model.

3.1.3 Loss Function

For each PHQ-8 item i and ordinal boundary k, the CORAL loss is:

$$\mathcal{L} = -\sum_{i=1}^{8}\sum_{k=1}^{3}\left[t_{i,k}\log\sigma(z_{i,k}) + (1-t_{i,k})\log(1-\sigma(z_{i,k}))\right], \tag{1}$$

where $t_{i,k} = \mathbb{1}[y_i \geq k]$ encodes whether the true severity exceeds boundary k. This cumulative formulation models the ordinal structure by predicting:

$$P(y_i \geq k) = \sigma(z_{i,k}). \tag{2}$$

3.1.4 Evaluation and Train Set Loss Across Input Levels

Table 3 summarizes the final losses across granularities, complementing the training and validation curves presented in Figures 2 and 3.

3.2 TextCNN

Convolutional Neural Networks (CNNs) were historically a dominant architecture for text classification and

Input-Level	Train Loss	Val Loss
Word-level	34.26	37.75
Sentence-level	32.40	38.21
Dialogue-level	37.24	35.11

Table 3: Final training and validation CORAL loss across input granularities.

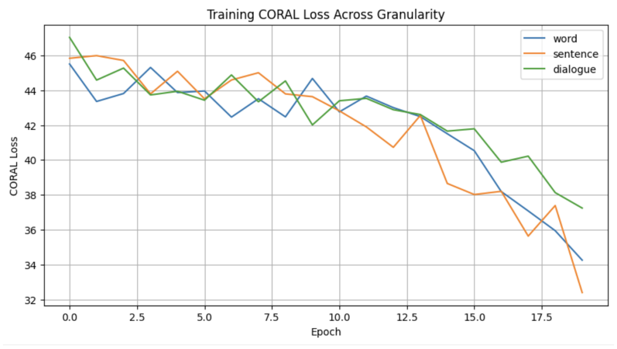


Figure 2: Training Dataset Loss Curve

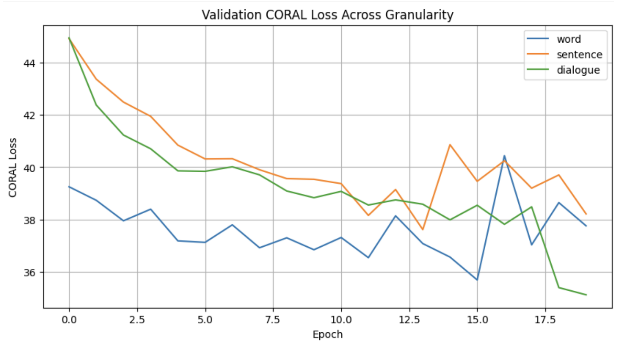


Figure 3: Validation Dataset Loss Curve

regression tasks due to their capacity to efficiently model local linguistic patterns. By applying multiple 1D convolutional filters, CNNs capture phrase-level features that compose into progressively higher-level abstractions across stacked layers. Relative to recurrent architectures of the same era, CNNs benefit from full parallelization across the sequence, enabling faster training while maintaining strong performance on tasks driven by local cues. However, these models inherently rely on fixed receptive fields determined by kernel sizes; as a result, modeling long-range, document-level dependencies becomes approximate rather than explicit, limiting their ability to capture broader semantic structure.

3.2.1 Architectural Summary

Under our formulation based on [3], the CNN processes each document through a sequence of convolutional feature extractors arranged in parallel. A token embedding layer first maps the input text into continuous vector representations. These embeddings are then passed through multiple convolutional blocks, each defined by a distinct kernel width and multiple learned filters, enabling the model to detect patterns spanning different n-gram lengths. The resulting feature maps are reduced via max-pooling operations, which select the most salient activation from each filter over the sequence. We concatenate all pooled outputs into a single fixed-dimensional representation, apply dropout for regularization, and project this vector through a fully connected layer to produce a scalar prediction for the document. As the task was a regression one, Mean Squared Error was used as the loss.

3.2.2 Training Configuration

Table 4 summarizes the key hyperparameters used to train the CNN model under all three input granularities.

Setting	Value
Batch size	64
Optimizer	AdamW
Learning rate	2×10^{-3}
Weight decay	1×10^{-5}
Gradient clipping	2.0
Epochs	10
Embedding dimension	100
Filters (per Kernel)	64
Kernel Sizes	[3,4,5,10]
Dropout	0.1

Table 4: Training configuration for the best-performing Text CNN model.

For model development, we first optimized the CNN architecture at the sentence-level granularity and subsequently applied the selected hyperparameters to the remaining dataset granularities to establish consistent baselines across conditions. During this tuning phase, we varied standard optimization parameters such as learning rate, dropout probability, and weight decay; however, architectural complexity emerged as the dominant factor influencing performance 5. Increasing the number of convolutional filters and the embedding dimensionality improved accuracy up to a point, with gains diminishing beyond 64 filters and approximately 100 embedding dimensions . Accordingly, we fixed these values for all downstream experiments.

	16 Filters	**32 Filters**	64 Filters
25 Embedding	36.30	29.14	24.70
50 Embedding	29.94	23.95	24.60
100 Embedding	27.44	22.40	20.73

Table 5: Validation MSE loss across different embedding and filter combinations

Kernel selection played a particularly influential role. While the absolute number of kernels mattered, the diversity of kernel sizes proved more consequential than their quantity alone. Initial experiments using kernel widths of 3, 4, and 5 exhibited stable performance, but introducing a substantially larger kernel (size 10) produced a marked improvement 6, suggesting that wider receptive fields help the model capture longer-range textual cues overlooked by narrowly focused filters. Surprisingly, larger receptive fields remained effective even at the word-level granularity, where locality effects should, in principle, be absent. This behavior is likely attributable to the model encountering a sufficiently broad range of lexical items within each sample, allowing the wider kernels to extract meaningful patterns from word co-occurrence rather than relying on sentence-level structure.

Kernel Sizes	**Validation MSE**
2,3,4	25.17
3,4,5	24.52
3,5,10	23.04
3,4,5,10	20.73

Table 6: Validation MSE loss across kernel sizes

We additionally explored the use of pretrained embeddings, evaluating the 2024 GloVe Gigaword 50-dimensional vectors[**?**] as a representative example. Although their performance was broadly comparable to that of learned embeddings, they introduced notable limitations. First, the embedding dimensionality is fixed at 50, which constrains the representational capacity; our earlier results indicated that increasing embedding size materially enhances predictive accuracy. Second, there is a domain mismatch between the Wikipedia-derived corpora—which emphasize factual, formal writing—and the more colloquial, symptom-oriented language present in our depression dataset. Thus, a production-ready system using task-specific learned embeddings would be expected to surpass pretrained performance when scaled appropriately.

3.2.3 Comparisons

Graphs were all generated from hyperparameters optimized on the sentence level granularity.

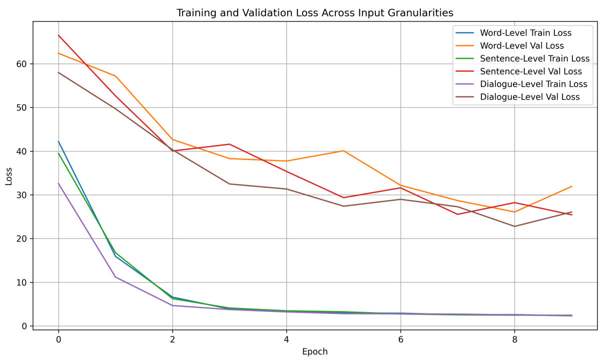


Figure 4: CNN Loss Curve Comparisons

Generally the training performance for all 3 granularity types were very similar, with a standard training curve in the first half of the training process, and validation instability in the second half. Notably, the validation instability is much more pronounced for the more granular input levels.

3.3 LSTM

3.3.1 Architectural Summary

The LSTM model serves as a recurrent baseline for comparison against the convolutional and attention-based architectures presented in previous sections. Each input sequence is first mapped into a trainable 128-dimensional embedding, followed by a two-layer unidirectional LSTM with hidden dimension 128 and dropout of 0.1. The hidden state from the final timestep is passed through a fully connected regression layer to produce the PHQ-8 score prediction. Due to its sequential processing nature, the LSTM is expected to be more sensitive to sequence length and long-range dependency modeling compared to attention-based architectures.

3.3.2 Hyperparameter Tuning and Training Configuration

A targeted hyperparameter search was performed using dialogue-level inputs prior to selecting the final LSTM configuration. We varied the hidden dimension, number of recurrent layers, and learning rate. The best-performing setting was then used for all input granular-

Table 7: Final tuned hyperparameters for the LSTM model.

Setting	Value
Max sequence length	512
Batch size	16
Embedding vocabulary	Trainable
Hidden dimension	128
Number of LSTM layers	2
Dropout	0.1
Output size	1
Optimizer	AdamW
Learning rate	$3e^{-4}$
Weight decay	$1e^{-5}$
Gradient clipping	1.0
Epochs	10
Random seed	2025

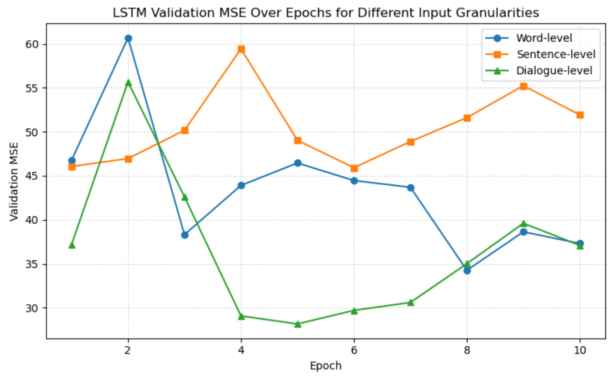


Figure 5: Validation MSE curves for the LSTM across word-, sentence-, and dialogue-level inputs.

ities (word, sentence, and dialogue) to ensure a controlled comparison focused on granularity rather than model capacity.

Table 7 summarizes the final hyperparameters used during training. All models were trained with MSE loss using the AdamW optimizer for 10 epochs, and gradient clipping was applied to stabilize training.

3.3.3 Training Behavior Across Granularities

Figures 5 and 6 illustrate the training behavior and participant-level prediction characteristics for the LSTM across the three input granularities. These figures are included here to provide a methodological view of the optimization dynamics and error distribution prior to reporting test performance in Section 4.4.

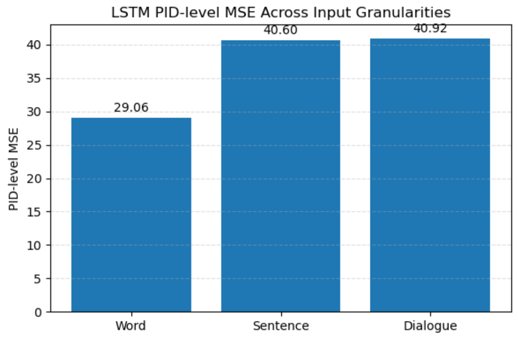


Figure 6: PID-level MSE comparison for the LSTM across input granularities.

3.4 Attention-RNN

3.4.1 Architectural Summary

For Attention-RNN implementation, we designed a two-level hierarchical encoder [2] designed to capture structure in PHQ-8 interview transcripts. At the lower text level (e.g. word level), each token is mapped to a trainable embedding vector and processed through a two-layer bidirectional GRU with dropout regularization. Then the vector-based attention module aggregates word-level representations into fixed-length sentence embeddings.

Next, the upper text level (e.g. sentence/dialogue level) applies another two-layer bidirectional GRU over the sequence of sentence embeddings. A second attention mechanism calculates sentence-level dynamics into a document vector that captures universal structure. Finally the final representation is normalized with LayerNorm, passed through dropout, and flow into a linear regression head producing a PHQ-8 score. The detailed step-by-step implementation of HANV2Attention network are provided below:

(1) Input. A dialogue is represented as S sentences, where sentence i contains T_i tokens:

$$x = \{x_{i,t} \mid i = 1, \ldots, S,\ t = 1, \ldots, T_i\}.$$

(2) Token Embedding. Each token is mapped to a $d_e = 300$ dimensional embedding, followed by normalization and dropout:

$$e_{i,t} = \text{Emb}(x_{i,t}) \in \mathbb{R}^{300}, \qquad \tilde{e}_{i,t} = \text{LayerNorm}(\text{Dropout}(e_{i,t})).$$

(3) Word-level Encoder. A bidirectional GRU with hidden size $h_w = 64$ per direction encodes word context:

$$\overrightarrow{h}_{i,t} = \overrightarrow{\text{GRU}}(\tilde{e}_{i,t}), \quad \overleftarrow{h}_{i,t} = \overleftarrow{\text{GRU}}(\tilde{e}_{i,t}),$$

$$h_{i,t} = [\overrightarrow{h}_{i,t};\ \overleftarrow{h}_{i,t}] \in \mathbb{R}^{128}.$$

(4) Word-level Attention. Sentence i aggregates its word representations using attention with intermediate size $d_a = 64$:

$$u_{i,t} = \tanh(W_w h_{i,t} + b_w), \quad W_w \in \mathbb{R}^{64\times128},\ b_w \in \mathbb{R}^{64},$$

$$\alpha_{i,t} = \frac{\exp(u_{i,t}^\top v_w)}{\sum_{t'} \exp(u_{i,t'}^\top v_w)}, \qquad v_w \in \mathbb{R}^{64},$$

$$s_i = \sum_{t=1}^{T_i} \alpha_{i,t} h_{i,t} \in \mathbb{R}^{128}.$$

(5) Sentence-level Encoder. A second bidirectional GRU with hidden size $h_s = 64$ per direction models inter-sentence structure:

$$\overrightarrow{g}_i = \overrightarrow{\mathrm{GRU}}(s_i), \quad \overleftarrow{g}_i = \overleftarrow{\mathrm{GRU}}(s_i),$$

$$g_i = [\overrightarrow{g}_i; \overleftarrow{g}_i] \in \mathbb{R}^{128}.$$

(6) Sentence-level Attention. The dialogue representation is computed by a second attention module:

$$v_i = \tanh(W_s g_i + b_s), \quad W_s \in \mathbb{R}^{64\times128},\ b_s \in \mathbb{R}^{64},$$

$$\beta_i = \frac{\exp(v_i^\top v_s)}{\sum_j \exp(v_j^\top v_s)}, \qquad v_s \in \mathbb{R}^{64},$$

$$d = \sum_{i=1}^{S} \beta_i g_i \in \mathbb{R}^{128}.$$

(7) Dialogue-level Pooling and Normalization. The dialogue vector is optionally passed through attention pooling and normalized:

$$h = \mathrm{LayerNorm}(d) \in \mathbb{R}^{128}.$$

3.4.2 Training Configuration

Models are optimized using AdamW with weight decay for stable convergence. Also, Mean squared error (MSE) is used. Hyperparameter configurations—embedding dimension, hidden size, hierarchical sequence limits, learning rate, weight decay, and dropout—are selected via grid search.

Table 8 summarizes the key hyperparameters used to train the Transformer model under all three input granularities. Also for simplicity and consistency, the grid search was operated given sentence-level text dataset to find the optimal training parameter as follows:

Setting	Value
Max words per sentence	100
Max sentences per document	30
Batch size	32
Optimizer	AdamW
Learning rate	1×10^{-5}
Weight decay	1×10^{-5}
Gradient clipping	1.0
Epochs	5
Embedding dimension	64
Hidden dimension (per GRU direction)	128
Number of GRU layers (word/sentence)	2
Attention type	Vector attention
Dropout	0.5

Table 8: Training configuration for the best-performing HAN-V2 model.

3.4.3 Evaluation and Train Set Loss Across Input Levels

We conduct independent experiments at three levels — word, sentence, and dialogue — by altering how the hierarchical encoder is fed text. The same HAN architecture is reused, while the dataset pipeline restructures transcripts into (i) single-sentence bags of words, (ii) sentence in documents, and (iii) full dialogue sequences. For each level, we deduce train and validation MSE to assess optimization behavior under different text levels.

Final test performance is reported as mean square error (MSE), reflecting the accuracy of aggregated predictions across all sentences belonging to the same participant. The figure provided displays the comparative MSE obtained for word-, sentence-, and dialogue-level inputs, illustrating how text level influences predictive stability and model sensitivity.

4 Results

4.1 Transformer Results

Figure 10 reports the final PID-level MSE of the Transformer model across the three input level. The PID-level metric represents the mean squared error between the model's predicted PHQ-8 total (obtained by summing the eight item predictions) and the ground-truth PHQ-8 score for each participant.

Across different input level, the Transformer achieves similar performance: word-level (41.90), dialogue-level (41.22), and a slightly higher error for sentence-level inputs (43.38). This suggests that the model remains relatively stable even when contextual structure is reduced.

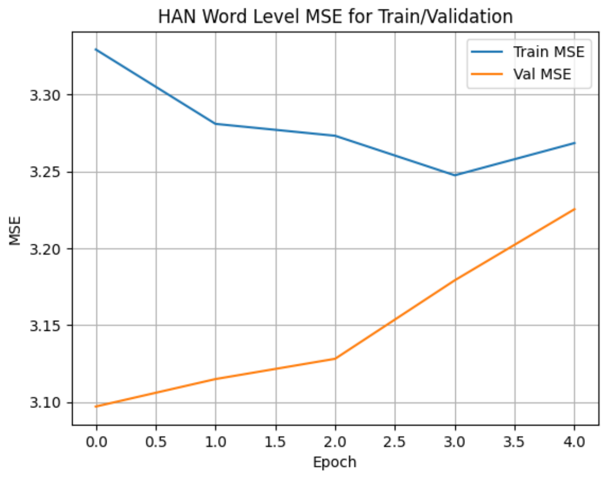


Figure 7: HAN MSE loss for train/validation dataset (word level)

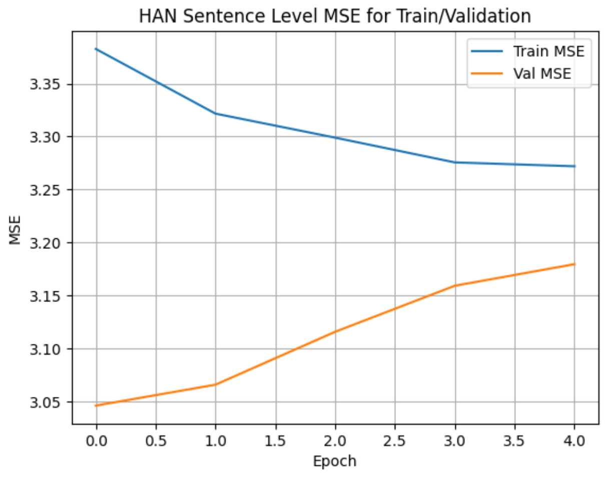


Figure 8: HAN MSE loss for train/validation dataset (sentence level)

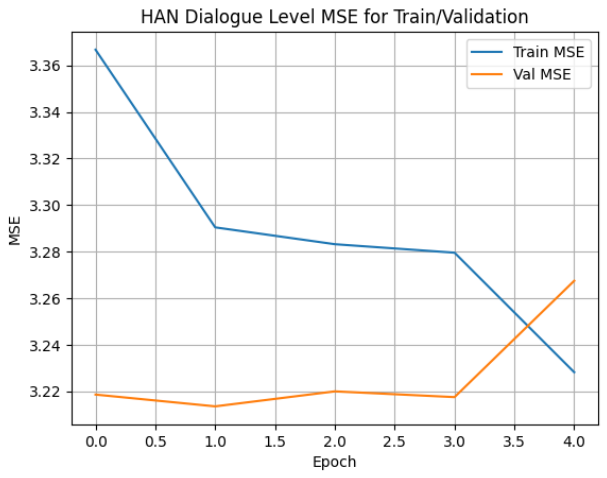


Figure 9: HAN MSE loss for train/validation dataset (dialogue level)

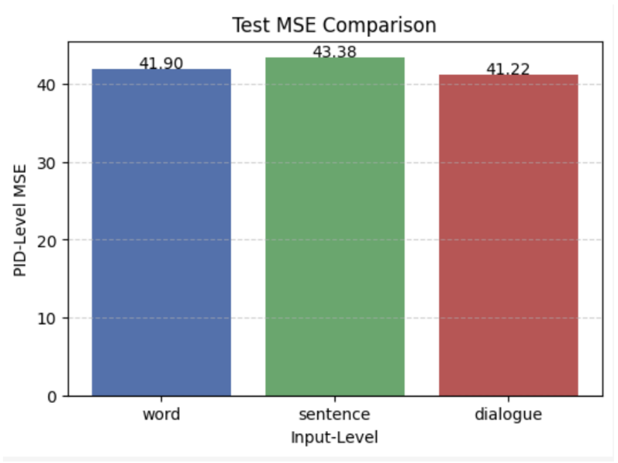


Figure 10: Test PID-level MSE across word-, sentence-, and dialogue-level inputs.

4.2 TextCNN Results

Table 9 reports the final PID-level MSE of the CNN model across the three input level. The PID-level metric represents the mean squared error between the model's predicted PHQ-8 total (obtained by summing the eight item predictions) and the ground-truth PHQ-8 score for each participant.

Across different input level, the CNN achieves similar performance on the test set: word-level (31.90), dialogue-level (31.04), and sentence-level inputs (30.77). This is contrasted with the variable performance on the validation sets. Since hyperparameter were selected by comparing their validation metrics, this difference in performance suggests mild overfitting on the validation set. Despite this, the stable test performance suggests that the model remains relatively stable even when contextual structure is reduced.

	Word Level	Sentence Level	Dialogue Level
Train MSE	2.25	2.02	2.23
Validation MSE	29.51	25.79	28.32
Test MSE	31.71	30.77	31.04

Table 9: Test PID Level MSE for CNN

4.3 Attention-RNN Results

Table 10 summarizes the final MSE of the hierarchical attention network (HAN) evaluated at the word-, sentence-, and dialogue-level inputs across the train, validation, and test splits. Consistent with prior work, the PID-level metric reflects the mean squared error between the predicted PHQ-8 total score—computed as the sum of the eight item-level outputs—and the corresponding ground-truth PHQ-8 total for each participant.

Overall, the model exhibits broadly comparable performance across the three input text level. On the test set, the HAN attains MSE values of 3.1270 (word-level), 3.0934 (sentence-level), and 3.1239 (dialogue-level), indicating minimal degradation when shifting between finer- and coarser-grained representations. The validation results display similarly close trends, although the slightly better validation MSE relative to training MSE suggests mild overfitting due to tuning hyperparameters on the validation set.

	Word Level	Sentence Level	Dialogue Level
Train MSE	3.2459	3.2828	3.2282
Validation MSE	3.2332	3.2256	3.2675
Test MSE	3.1270	3.0934	3.1239

Table 10: Test PID Level MSE for CNN

4.4 LSTM Results

Table 11 summarizes the final train, validation, and PID-level MSE across all three input granularities. The LSTM displays a clear preference for word-level inputs, achieving the lowest PID-level MSE (29.06). In contrast, both the sentence-level (40.60) and dialogue-level (40.92) variants exhibit substantially higher error. This pattern aligns with the training curves shown in Figures 5 and 6, where the sentence-level model consistently demonstrates unstable validation behavior and higher overall loss. The dialogue-level model, while showing more stable convergence, does not translate this advantage into improved participant-level prediction.

These results suggest that the LSTM benefits most from fine-grained lexical information, and its limited ability to model long-range dependencies may hinder performance when processing longer sentence- and dialogue-level sequences. This contrasts with the Transformer model (Section 4.1), which remains more robust across granularities due to its attention-based contextual encoding.

Table 11: LSTM performance across input granularities. Train and validation metrics are sample-level MSE, while the test metric is PID-level MSE.

	Word Level	Sentence Level	Dialogue Level
Train MSE	1.6122	1.2743	0.6567
Validation MSE	37.2986	51.9317	37.0843
PID MSE	29.0630	40.6029	40.9237

5 Conclusion

Across four architectures (CNN, LSTM, Attention-RNN, and Transformer), our experiments show that varying the textual granularity of the input—word, sentence, or dialogue level—does not produce substantial or systematic differences in PHQ-8 regression performance. Despite theoretical expectations that higher contextual levels should yield stronger affective signal, the empirical results indicate that, within the constraints of the DAIC-WOZ dataset and the models studied, input granularity alone does not fundamentally alter predictive accuracy.

6 Discussion

While our experiments compared three levels of textual granularity—word-level tokens, sentence-level units, and full-dialogue inputs—the results should be interpreted with caution. The DAIC-WOZ dataset itself contains known sources of variability, including interviewer effects, inconsistent transcription quality, and imbalanced PHQ-8 severity distributions. These factors may introduce biases that mask subtle differences between granularities. In addition, model training is sensitive to random seeds, optimization dynamics, and subjective design choices (e.g., segmentation rules, truncation thresholds), any of which could influence the outcomes reported here.

Nevertheless, the overall stability across input granularities suggests a deeper possibility: modern architectures may already be extracting the dominant affective cues present in the text, regardless of granularity. This raises the question of whether future improvements will depend less on input quality or segmentation resolution, and more on leveraging stronger representation learning—such as transfer learning with large pretrained language models, hierarchical transformers, or multimodal alignment of acoustic and lexical cues. Such models may be capable of capturing higher-order correlations that simpler architectures cannot disentangle within this dataset.

7 Team Contributions

Author	**Contributions**
Haike Yu	Transformer algorithm and experiments;
	Designed data web-scraping scripts;
	Built preprocessing pipeline;
	Organized the codebase structure;
	Outlined the paper structure;
	Served as team lead.
Sun Bin Mun	Implemented and evaluated the Attention RNN model.
Edbert Wang	Implemented and evaluated the CNN model.
Chengyuan Yao	Implemented and evaluated the LSTM model.

8 Github Link

Github: https://github.com/RemMyFav/DeepPHQ

References

[1] Distress analysis interview corpus – wizard of oz (daic-woz) dataset, 2014. 189 clinical interviews with audio, video, transcripts, and PHQ-8 labels.

[2] Dongping Fang, Lian Duan, Xiaojing Yuan, Allyn Klunder, Kevin Tan, Suiting Cao, Yeqing Ji, and Mike Xu. Interpretable hierarchical attention network for medical condition identification, 2024.

[3] Hongxia Lu, Louis Ehwerhemuepha, and Cyril Rakovski. A comparative study on deep learning models for text classification of unstructured medical notes with various levels of class imbalance. *BMC Medical Research Methodology*, 22(1):181, 2022.