QoSBERT: An Uncertainty-Aware Approach Based on Pre-trained Language Models for Service Quality Prediction

Ziliang Wang, Xiaohong Zhang, Ze Shi Li and Meng Yan, Member, IEEE

Abstract—Accurate prediction of Quality of Service (QoS) metrics is fundamental for selecting and managing cloud-based services. Traditional QoS models rely on manual feature engineering and yield only point estimates, offering no insight into the confidence of their predictions. In this paper, we propose QoSBERT, the framework that reformulates QoS prediction as a semantic regression task based on pre-trained language models. Unlike previous approaches relying on sparse numerical features, QoSBERT automatically encodes user-service metadata into natural language descriptions, enabling deep semantic understanding. Furthermore, we integrate a Monte Carlo Dropout–based uncertainty estimation module, allowing for trustworthy and risk-aware service quality prediction, which is crucial yet underexplored in existing QoS models. QoSBERT encodes user-service metadata as natural language and leverages a pre-trained model to capture contextual semantics. It applies attentive pooling over the encoded embeddings and employs a lightweight regressor optimized to minimize prediction error. To quantify predictive confidence, Monte Carlo Dropout is applied at inference time. The resulting uncertainty estimates further support high-confidence sample selection, enhancing robustness in low-resource scenarios. On standard QoS benchmark datasets, QoSBERT achieves an average reduction of 11.7% in MAE and 6.7% in RMSE for response time prediction, and 6.9% in MAE for throughput prediction compared to the strongest baselines, while providing well-calibrated confidence intervals for robust and trustworthy service quality estimation. Our approach not only advances the accuracy of service quality prediction but also delivers reliable uncertainty quantification, paving the way for more trustworthy, data-driven service selection and optimization.

Index Terms—Service recommendation, QoS prediction, LLM, Uncertainty aware

1 Introduction

In recent times, various domains and applications, such as cloud services, online streaming, and e-commerce, are increasingly being offered as a service, resulting in a greater emphasis on ensuring optimal Quality of Service (QoS) [1], [2], [3]. QoS values are frequently used as crucial inputs for various downstream service computing tasks, such as service recommendation [4], [5] and service composition [6], [7]. At the same time, some microservice optimization techniques require accurate predictive Quality of Service (QoS) models, such as autoscaling based on QoS [8], [9]. Consequently, accurately predicting QoS has become a critical topic in service computing [10], [11], [12].

Existing QoS prediction methods can be broadly categorized into two major paradigms: collaborative filtering-based approaches and deep learning-based approaches. Collaborative filtering methods are typically further divided into memory-based and model-based techniques. Memory-based approaches rely on historical QoS interaction data, computing similarity measures among users or services to form neighborhood-based predictions for unobserved QoS values [10], [13], [14]. Despite their straightforward implementation, memory-based approaches often encounter significant limitations due to data sparsity. On the other hand, model-based collaborative filtering addresses sparsity by deriving latent semantic representations from QoS interaction histories to identify underlying relationships between users and services [15]. However, traditional model-based techniques generally capture only linear or simple non-linear interactions, limiting their effectiveness in capturing complex and high-dimensional semantic relationships inherent in QoS data. To address this issue, recent studies have explored advanced deep learning techniques [16], [17], utilizing neural network architectures such as multilayer perceptrons to model sophisticated, nonlinear user-service interactions, substantially improving representational richness and predictive accuracy compared to conventional collaborative filtering approaches.

However, accurately predicting QoS is significantly hindered by two major challenges: the underutilization of semantic-rich service and user descriptions, and the lack of uncertainty estimation, both of which limit the robustness and expressiveness of existing models in real-world scenarios.

First, although modern service platforms often provide rich textual descriptions of user contexts and service functionalities, most QoS prediction models rely on sparse numerical features or interaction matrices, failing to capture the inherent semantics embedded in these descriptions [16], [18], [19], [20]. As a result, existing models often struggle to generalize across heterogeneous services or users, especially when faced with cold-start scenarios

- *Ziliang Wang is with Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China and School of Big Data and Software Engineering, Chongqing University, Chongqing, China and also with the Key Lab of HCST (PKU), MOE; SCS, Peking University, China.*
  Xiaohong Zhang, Meng Yan are with Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China and School of Big Data and Software Engineering, Chongqing University, Chongqing 401331, China.
- *Ze Shi Li is an assistant professor in Computer Science at the University of Oklahoma, USA.*
- *Ziliang Wang is the corresponding author.*
  E-mail: wangziliang@pku.edu.cn

or sparse access records. For example, Zhang et al. encoded the user location information into an integer value through the encoding layer [18].

Second, current models typically generate deterministic QoS predictions without quantifying prediction confidence, making it difficult to distinguish between reliable and potentially risky estimates. This limitation poses a serious concern for practical service selection and adaptation tasks, where accurate confidence estimation is essential for mitigating risks, especially under distribution shifts or noisy inputs. Ye et al. proposed CMF to discuss the study of data noise in QoS tasks, but the research on the trustworthiness of prediction models is still blank [21]. Therefore, there is a pressing need to explore models that can (1) effectively encode natural language representations of services and users, and (2) produce well-calibrated QoS predictions with uncertainty awareness.

To address these challenges, we propose QoSBERT, a novel approach that leverages pre-trained language models and uncertainty estimation to improve the accuracy and robustness of QoS prediction. QoSBERT introduces three innovative components to tackle the aforementioned limitations: (1) a semantic-aware representation framework based on pre-trained language models, (2) a Monte Carlo dropout mechanism for predictive uncertainty estimation, and (3) an attention-enhanced pooling module for refined feature aggregation.

QoSBERT fills the gap in this field by proposing a prediction architecture based on a pre-trained model. First, unlike conventional methods that rely on sparse numerical features or handcrafted vectors, QoSBERT converts user and service attributes into structured textual templates and encodes them using powerful pre-trained models (e.g., Qwen2.5 [22]). This enables the model to effectively transform structured user–service metadata into contextualized semantic representations, offering a practical alternative to manual feature engineering and sparse numerical encoding.

Second, to enhance prediction reliability, QoSBERT introduces a Monte Carlo dropout mechanism during inference. By stochastically sampling multiple forward passes through dropout-enabled layers, the model is able to estimate the variance of predictions and quantify the epistemic uncertainty inherent in sparse QoS datasets. This uncertainty signal is particularly useful for identifying low-confidence predictions and mitigating overconfident failures. Third, we integrate a lightweight attention-based pooling module that fuses token-level representations and enables the model to adaptively focus on semantically critical components in the input sequence.This further boosts the model's ability to learn fine-grained features relevant to QoS outcomes, even under noisy or long-tail distributed conditions.

In summary, this paper makes the following key contributions:

a) We propose QoSBERT, a novel high-precision framework that reformulates QoS prediction as a semantic regression task over structured service metadata. By leveraging pre-trained language models, our approach enables effective modeling of descriptive user–service interactions without relying on manual feature engineering or domain heuristics.

b) We design a Monte Carlo dropout-based uncertainty estimation mechanism that enables the model to quantify predictive confidence and improves robustness under sparse or noisy conditions.

c) We conduct extensive experiments on standard QoS benchmarks (e.g., WS-DREAM) and demonstrate that QoSBERT consistently outperforms state-of-the-art baselines, achieving an average reduction of 11.7% in MAE and 6.7% in RMSE for response time prediction, and 6.9% in MAE for throughput prediction. In addition, QoSBERT offers well-calibrated uncertainty estimates, enhancing the interpretability and reliability of QoS-aware decision making. To facilitate reproducibility and future research, we have released the full source code, datasets, and data preprocessing scripts for QoSBERT[1].

The remainder of this paper is organized as follows: Section 2 reviews related work on QoS prediction, with emphasis on collaborative filtering, deep learning, and federated approaches. Section 3 presents the proposed QoSBERT framework in detail, including its semantic-aware feature construction, transformer-based encoding, and uncertainty-aware regression head. Section 4 describes the experimental setup, including dataset configurations, evaluation metrics, baseline models, and implementation details. Section 5 reports the experimental results and performance comparisons across different QoS metrics and data sparsity levels. Section 6 concludes the paper and outlines future directions, such as model compression and extension to multi-modal QoS scenarios.

2 Related Work

Collaborative Filtering for QoS Prediction. Collaborative filtering (CF) has been widely employed for Quality of Service (QoS) prediction due to its ability to capture behavioral patterns among users and services based on historical interactions [23], [24], [25], [26]. Traditional CF-based methods can be broadly categorized into two families: memory-based and model-based approaches [27]. Memory-based CF directly estimates unknown QoS values by measuring similarity between users or services, often relying on features such as geographic location, response time profiles, or network attributes. Representative examples include user-based methods like UPCC [28], service-oriented techniques like IPCC [27], and hybrid extensions such as UIPCC [29] that jointly consider user-service pairs. These techniques are computationally efficient and simple to implement, but tend to suffer from poor scalability and limited modeling capacity in sparse environments.

In contrast, model-based CF learns latent representations of users and services through training on historical QoS matrices, enabling better generalization and handling of data sparsity. Matrix factorization (MF) is one of the most prominent model-based paradigms [30], [31], where latent vectors are optimized to approximate observed QoS entries. Further enhancements integrate auxiliary information such as time [32], location [14], or trust relationships to refine prediction. To address overfitting and interpretability, several variants introduce constraints or regularization; for instance, Luo et al. proposed a non-negative latent factor model (NLF) that incorporates structural assumptions into latent space [33]. Despite their predictive power, both memory- and model-based CF approaches often rely on centralized data collection, raising concerns about privacy leakage and making them less suitable for privacy-sensitive or distributed service environments.

Deep Learning for QoS Prediction. In recent years, deep learning techniques have been increasingly adopted to address the limitations of traditional collaborative filtering in modeling non-linear user-service relationships. Neural network-based collaborative filtering (NCF) [34] pioneered this direction by replacing the

1. The code and data is available for reproduction: https://github.com/HotFrom/QoSBERT

inner product with a trainable neural architecture, allowing more expressive interaction modeling. To further incorporate temporal and contextual patterns, various time-aware deep models have been proposed. For instance, Zhou et al. [35] encoded temporal dynamics by associating latent variables with discrete time intervals, while Wang et al. [36] captured sequential dependencies using dynamic Bayesian structures. Other efforts integrate multiple modalities; DHSR [37] fused text semantics and neural networks to infer complex relations between services and mashups, improving recommendation performance. More recently, graph neural networks (GNNs), residual connections, and deep latent state models have been explored to better leverage structured and implicit information in sparse QoS scenarios [38], [39], [40].

Beyond modeling accuracy, recent studies have increasingly emphasized trustworthiness and data privacy in QoS prediction. For example, FRLN [41] introduces a residual ladder network within a federated learning framework to extract multi-level latent features while safeguarding user data through personalized local training. Similarly, PE-FGL [42] leverages privacy-enhancing graph construction and secure federated aggregation to enable accurate and confidential QoS estimation across distributed clients. Liu et al. propose llmQoS [43], which simply converts user- and service-side attributes into descriptive text and fine-tunes a large language model without any task-specific adaptations for data sparsity or numerical regression, so its prediction accuracy lags behind hybrid CF approaches and uncertainty-aware models designed explicitly for QoS prediction. Recently, Google reported a text-to-text Regression Language Model for performance prediction on large industrial systems (e.g., Borg), showing competitive accuracy [44];

3 Approach

The architecture of QoSBERT, as illustrated in Fig. 1, consists of four primary stages: feature description construction, contextual fusion via pre-trained models, uncertainty-aware prediction using Monte Carlo Dropout, and calibrated confidence estimation via temperature scaling.

3.1 Feature Description Construction

Traditional QoS prediction models typically encode users and services using discrete IDs or sparse numerical features, which severely limits their ability to capture the underlying semantics of user-service interactions. These handcrafted or categorical features fail to express contextual relationships such as geographic proximity, network similarity, or routing topologies.

To address this limitation, we propose a natural language feature construction strategy that reformats structured metadata (e.g., IP address, AS number, country, region) into unified textual templates. While not fully unstructured, these template-based textual features descriptions provide contextualized cues that can be effectively consumed by pre-trained language models.

Each QoS sample is initially defined as a triplet:

$$L = (U, S, T) \tag{1}$$

where U and S represent the user and service metadata respectively, and T is the observed QoS value (e.g., response time or throughput).

We implement a rule-based parser that extracts structured fields—such as `UserID`, `IP address`, `AS number`, and `Country`—from annotated data, and rephrases them into a unified template-based natural language form.

This string is then stored under the `"feature"` field in the JSONL data format, replacing the original raw code or sparse numerical representation. The transformation is implemented through a script that uses regular expressions to extract field values and format them into the template above, as shown in our data preprocessing pipeline. This textual encoding provides two key advantages:

1) It allows seamless integration with pre-trained language models (e.g., RoBERTa [45], CodeGen [46],Qwen2.5 [22]), which are naturally designed to process tokenized text and leverage semantic priors from large-scale corpora.
2) It enriches the sparse categorical metadata by projecting it into dense, contextualized embeddings, enabling better generalization in cases such as cold-start scenarios or rare service-user pairs.

These tokenized natural language inputs are then passed to the Transformer encoder for further contextual feature fusion, which we elaborate in the next subsection.

3.2 Fusion Modeling with Pre-trained Models

Following the construction of semantic feature descriptions, we encode the user-service textual pairs using a pre-trained Transformer model (e.g., CodeGen or Qwen2.5-0.5B-Instruct) to obtain contextualized token representations. Each textual input is first tokenized into subword units using the corresponding tokenizer of the backbone model, transforming the raw description into a sequence of discrete token IDs. Formally, given a user description U and a service description S, the input sequence is structured as:

$$X = [\texttt{[BOS]}, U_0, \ldots, U_k, \texttt{[SEP]}, S_0, \ldots, S_n, \texttt{[EOS]}]$$

where `[BOS]` and `[EOS]` denote the beginning and end of the sequence, and `[SEP]` separates user and service parts.

The tokenized input X is then passed through the Transformer encoder, yielding a set of hidden states:

$$H = \{h_1, h_2, \ldots, h_L\} \in \mathbb{R}^{L \times d}$$

where L is the sequence length and d is the hidden dimensionality.

(1) Multi-Layer Feature Fusion. Recent empirical studies [45], [47] have shown that different layers in Transformer models encode different levels of linguistic and semantic information. Lower layers typically capture local syntactic patterns, while deeper layers represent more abstract and task-specific features. However, relying solely on the last layer may lead to overfitting on noisy or unstable features, especially when the downstream dataset is small. Therefore, to balance generality and task-awareness, we aggregate the representations from the last K encoder layers (default $K = 4$) to form a fused contextual feature:

$$H^{\text{fused}} = \frac{1}{K} \sum_{i=N-K+1}^{N} H^{(i)}$$

where N is the total number of Transformer layers, and $H^{(i)}$ is the hidden state output at layer i.

This layer fusion strategy not only smooths potential noise from individual layers but also captures richer semantic cues for QoS prediction.

(2) Multi-pooling. We further apply three pooling strategies on H^{fused} to extract global sequence features:

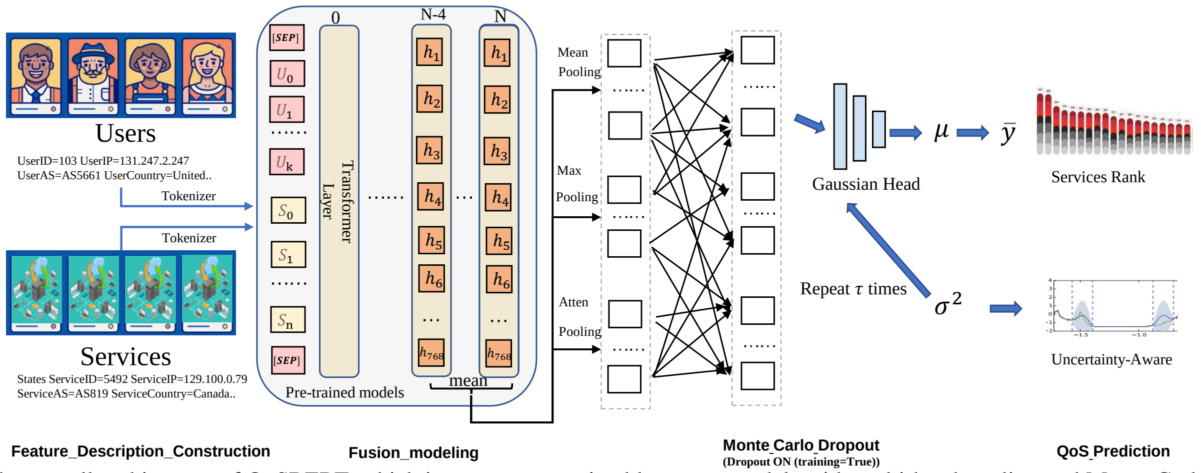


Fig. 1: The overall architecture of QoSBERT, which integrates pre-trained language models with multi-head pooling and Monte Carlo dropout to enable uncertainty-aware QoS prediction based on user-service textual descriptions.

Algorithm 1 QoSBERT: QoS Prediction with Uncertainty Estimation

```
Require: QoS records $L = \{(U_i, S_i)\}_{i=1}^{N}$; dropout iterations $T$; temperature parameter $\tau$
Ensure: Predicted QoS $\mu_i$ and calibrated variance $\sigma^2_{i,\mathrm{cal}}$
 1: // Feature Construction
 2: for $i = 1$ to $N$ do
 3:     Generate input $x_i = \mathtt{desc}(U_i) \| \mathtt{desc}(S_i)$
 4: end for
 5: // Contextual Encoding
 6: for $i = 1$ to $N$ do
 7:     Encode $x_i$ using a pretrained Transformer to get layer outputs $H_i^{(1)}, \ldots, H_i^{(L)}$
 8:     Fuse top-$K$ layers: $H_i^{\mathrm{fused}} = \frac{1}{K} \sum_{l=L-K+1}^{L} H_i^{(l)}$
 9:     Apply mean, max, and attention pooling:
            $v_i = \mathrm{Mean}(H_i^{\mathrm{fused}}) \| \mathrm{Max}(H_i^{\mathrm{fused}}) \| \mathrm{Attn}(H_i^{\mathrm{fused}})$
10: end for
11: // Monte Carlo Dropout Sampling
12: for $t = 1$ to $T$ do
13:     For all $i$, apply dropout and predict: $(\mu_i^{(t)}, \sigma_i^{2(t)}) = f_\theta^{(t)}(v_i)$
14: end for
15: Compute predictive mean:
            $\mu_i = \frac{1}{T} \sum_{t=1}^{T} \mu_i^{(t)}$
16: Compute predictive variance:
            $\sigma_i^2 = \underbrace{\frac{1}{T} \sum_{t=1}^{T} \sigma_i^{2(t)}}_{\text{aleatoric}} + \underbrace{\frac{1}{T} \sum_{t=1}^{T} \left(\mu_i^{(t)}\right)^2 - \mu_i^2}_{\text{epistemic}}$
17: // Temperature Scaling (Inference Only)
            $\sigma^2_{i,\mathrm{cal}} = \tau^2 \cdot \sigma_i^2$
18: return $\mu_i$, $\sigma^2_{i,\mathrm{cal}}$
```

- **Mean Pooling:** element-wise mean over all tokens.
- **Max Pooling:** element-wise maximum over all tokens.
- **Attention Pooling:** a learnable self-attention pooling module to capture task-specific token contributions.

The three pooled vectors are added to form the final sequence embedding:

$$\mathbf{v}_{\mathrm{seq}} = \mathrm{Mean}(H^{\mathrm{fused}}) + \mathrm{Max}(H^{\mathrm{fused}}) + \mathrm{Attn}(H^{\mathrm{fused}})$$

This vector $\mathbf{v}_{\mathrm{seq}}$ represents the joint semantics of user and service, and is used as input for the downstream QoS regression module described in the next subsection.

3.3 Uncertainty-Aware QoS Prediction

3.3.1 *Monte Carlo Dropout and Variance Estimation*

To estimate both the expected QoS value and its predictive uncertainty, we adopt a Bayesian regression strategy based on Monte Carlo Dropout (MC Dropout) [48]. During inference, we perform T stochastic forward passes with dropout layers *activated* and obtain T pairs of Gaussian parameters $(\mu^{(i)}, \sigma^{2(i)})$:

$$(\mu^{(i)}, \sigma^{2(i)}) = \mathscr{F}_\theta^{(i)}(\mathbf{v}_{\mathrm{seq}}), \qquad i = 1, \ldots, T, \tag{2}$$

where $\mathscr{F}_\theta^{(i)}$ denotes the i-th stochastic instantiation of the network.

Predictive mean:The final point estimate of QoS is the sample mean:

$$\mu = \frac{1}{T} \sum_{i=1}^{T} \mu^{(i)}. \tag{3}$$

Predictive variance: Following the law of total variance, the overall predictive variance is decomposed into *aleatoric* (data-level) and *epistemic* (model-level) uncertainty:

$$\sigma^2_{\mathrm{ale}} = \frac{1}{T} \sum_{i=1}^{T} \sigma^{2(i)}, \tag{4}$$

$$\sigma^2_{\mathrm{epi}} = \frac{1}{T} \sum_{i=1}^{T} (\mu^{(i)})^2 - \mu^2, \tag{5}$$

$$\sigma^2 = \sigma^2_{\mathrm{ale}} + \sigma^2_{\mathrm{epi}}. \tag{6}$$

This formulation captures both the intrinsic data noise and the model's uncertainty due to limited knowledge (e.g., small training data or ambiguous inputs).

3.3.2 *Temperature Scaling for Calibration*

Although MC Dropout provides a principled uncertainty estimate, the raw variance may still be miscalibrated. We therefore introduce a learnable *temperature* parameter $\tau > 0$ to rescale the predicted total variance:

$$\sigma^2_{\text{cal}} = \tau^2 \cdot \sigma^2, \qquad \tau = \exp(\log \tau), \tag{7}$$

where $\log \tau$ is optimized as a scalar parameter on a held-out calibration set by minimizing the negative log-likelihood (NLL) of the Gaussian predictive distribution:

$$\mathscr{L}_{\text{NLL}} = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{(y_i - \mu_i)^2}{2\sigma^2_{\text{cal},i}} + \frac{1}{2}\log \sigma^2_{\text{cal},i}\right]. \tag{8}$$

The calibrated variance σ^2_{cal} is subsequently used for downstream risk-aware QoS prediction and decision-making. The y_i represents the true label.

3.3.3 *Final Objective*

During training, the model jointly minimizes the negative log-likelihood and the Mean Absolute Error (MAE) to balance probabilistic calibration and point-wise accuracy:

$$\mathscr{L} = (1-\alpha)\cdot \mathscr{L}_{\text{NLL}} + \alpha \cdot \mathscr{L}_{\text{MAE}}, \quad \mathscr{L}_{\text{MAE}} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \mu_i| \tag{9}$$

Here, α is a hyperparameter that controls the trade-off between calibration and prediction precision. We use α = 0.9 in all reported results; nearby values (±0.05) yielded similar performance, while α = 0.9 gave the best average MAE/RMSE across densities. At inference time, we report both the predicted mean μ and calibrated uncertainty σ^2_{cal} to support risk-aware decision making in QoS-sensitive applications.

3.4 Implementation details

TABLE 1: Hyperparameters used for training QoSBERT.

Hyperparameter	**Value**
Model type	Qwen (2.5-0.5B-Instruct)
MC Dropout passes (T)	20
Dropout probability	0.1
Block size	64
Batch size	64
Learning rate λ	2×10^{-5}
Max gradient norm	1.0
Random seed	42

To further enhance the learning capability of the encoder and enable more task-specific adaptation, we adopt a gradual layer unfreezing strategy inspired by domain-adaptive pretraining techniques [49].

Instead of fine-tuning the entire pretrained model at once, which may lead to catastrophic forgetting or overfitting on small-scale QoS data, we initially freeze all encoder parameters except for LayerNorm and Embedding layers. This guarantees stable gradient propagation and preserves general knowledge encoded in the backbone model. Specifically, we adopt Qwen2.5-0.5B-Instruct as our backbone model [50] as shown in Table 1, which is a general-purpose, instruction-tuned causal language model with approximately 0.5 billion parameters. Its architecture includes Transformer blocks enhanced by Rotary Position Embeddings (RoPE), SwiGLU activations, RMSNorm normalization, Attention with QKV bias, and tied word embeddings. The model supports context lengths up to 32,768 tokens and can generate sequences up to 8,192 tokens, enabling efficient semantic understanding and representation of diverse textual data.

During training, the model progressively unfreezes the top-most n layers of the Transformer encoder in every k epochs, allowing deeper semantic alignment with the QoS prediction task while avoiding early instability. Let $\mathscr{L}_{\text{encoder}} = \{\ell_1, \ell_2, ..., \ell_L\}$ denote the list of all encoder blocks, our approach incrementally activates gradient updates for the top layers:

$$\mathscr{L}^{(e)}_{\text{active}} = \{\ell_{L-n(e)+1}, ..., \ell_L\} \tag{10}$$

where $n(e)$ is the number of unfreezed layers at epoch e, controlled by a linear schedule.

This strategy offers two advantages:

1) It enables the model to retain generalizable features from pre-training while gradually adapting to domain-specific patterns in service quality data.
2) It stabilizes the optimization landscape in early epochs and improves convergence in the later stage by carefully exposing trainable parameters.

3.5 Complexity Analysis

In this subsection, we analyze the computational complexity of QoSBERT during both training and inference stages.

Training Complexity. During training, each input description is tokenized and fed into a pre-trained Transformer encoder. The dominant computational cost stems from the Transformer forward pass, which scales as:

$$\mathscr{O}(L^2 d)$$

where L is the input sequence length and d is the hidden dimension size. Following the encoder, the multi-view pooling (mean, max, attention) operations are linear in L, and the subsequent regression head introduces negligible additional complexity compared to the encoder backbone.

Moreover, the gradual unfreezing strategy ensures that in the early epochs, only a small subset of layers are updated, effectively reducing training overhead during the initial optimization phase.

Inference Complexity. In the inference phase, QoSBERT applies Monte Carlo Dropout to perform T stochastic forward passes for uncertainty estimation. Therefore, the total inference cost is approximately T times that of a standard single forward pass:

$$\mathscr{O}(T \times (L^2 d))$$

where T is typically set to a moderate value (e.g., $T = 20$) to balance between predictive performance and efficiency.

Since only the regression head is repeatedly applied after the encoder outputs are obtained, and considering modern GPU parallelization capabilities, this overhead remains acceptable for most QoS prediction scenarios. In Table 1, we have described the default model used in this paper and the core training parameters.

Overall, QoSBERT introduces moderate additional computational overhead compared to standard Transformer-based regression models due to the use of MC Dropout. Specifically, we train on three A100 GPUs, each occupying about 25 GB of video memory for large-batch optimization. Running QoSBERT in offline-inference mode (T=20) on three NVIDIA A100 (40 GB) GPUs achieves an *aggregate throughput of* 231.21 *samples* s^{-1}, with a *mean batch latency of* 1091.31 *ms* and *sample-level latency of* 4.28 *ms*. The latency distribution is stable, with P50 = 1091.36 ms and P95 = 1099.26 ms. These results suggest that although QoS-BERT employs MC Dropout with T=20 forward passes, the inference time remains acceptable for many non-real-time or batch-oriented QoS prediction tasks. Although the batch-level latency reaches 1099 ms due to large batch size (e.g., 256 samples per batch) and 20 MC Dropout passes per sample, the per-sample latency remains as low as 4.28 ms. Moreover, the batch latency distribution is highly stable (P50 $\approx$ P95), suggesting minimal runtime variance and excellent inference consistency. This makes QoSBERT well-suited for batch-mode QoS prediction in service orchestration pipelines. This overhead is justified by the significant gains in predictive uncertainty estimation and model robustness, which are critical for reliable QoS-sensitive service computing applications.

Inference optimization strategies. Although QoSBERT adopts MC Dropout for uncertainty-aware prediction, the inference cost can be substantially reduced in practice through several optimization strategies. First, the number of Monte Carlo samples T can be adjusted based on resource constraints: our ablation in Section 5.3 shows that even with $T = 5$, QoSBERT retains most of its accuracy benefits while reducing inference time by over 75%. Second, for latency-sensitive settings, the model can be deployed with smaller batch sizes or on a single GPU with limited degradation. Finally, future work can incorporate model compression techniques such as quantization or distillation to further reduce memory usage and computational load. These flexible configurations make QoSBERT applicable to both high-throughput and resource-constrained service environments.

4 STUDY SETUP

TABLE 2: Division of dataset under different sparsity levels.

No.	Density	Split (Train:Test:Valid)	Train	Test	Validation
RT:D1.1	0.05	5%:75%:20%	95,877	1,437,361	383,310
RT:D1.2	0.10	10%:70%:20%	191,755	1,341,483	383,310
RT:D1.3	0.15	15%:65%:20%	287,632	1,245,606	383,310
RT:D1.4	0.20	20%:60%:20%	383,510	1,149,728	383,310
TP:D2.1	0.05	5%:75%:20%	82,586	1,218,788	330,343
TP:D2.2	0.10	10%:70%:20%	165,171	1,136,203	330,343
TP:D2.3	0.15	15%:65%:20%	247,757	1,053,617	330,343
TP:D2.4	0.20	20%:60%:20%	330,343	971,031	330,343

4.1 Datasets

We evaluate the proposed QoSBERT model on a refined version of the widely used WS-Dream dataset [25]. WS-Dream contains Quality of Service (QoS) records collected from 339 users invoking over 5,800 real-world web services across various geographical regions. Each invocation record includes two types of QoS attributes: response time (RT, in seconds) and throughput (TP, in kbps). To better align the data with the modeling assumptions of QoSBERT and enhance prediction reliability, we preprocess the original dataset through the following steps:

(1) Semantic Feature Construction. Each QoS record originally consists of basic invocation information and limited numerical features regarding users and services. As shown in Table 3, to enable semantic understanding, we enrich these records by retrieving additional metadata from the user and service tables, such as IP addresses, Autonomous Systems (AS), geographic locations, and WSDL providers. These fields are then assembled into structured natural language templates, which serve as inputs to the Transformer encoder. This textual construction not only provides rich contextual cues for model learning but also bridges the gap between sparse numerical features and pre-trained language models' requirements. The data of this paper built by this method is shared in the open source package.

TABLE 3: Original Data Structure of QoS Records, Users, and Services.

Field	Description
User Metadata	
IP Address	Public IP address of the user.
Country	Country derived from IP geolocation.
IP Number	Numeric representation of the IP.
AS	Registered Autonomous System name and number.
Latitude/Longitude	Geolocation coordinates.
Service Metadata	
WSDL Address	URL of the service description (WSDL).
Service Provider	Hosting organization.
IP Address	Public IP address of the service server.
Country	Service server location by country.
IP Number	Numeric representation of the service IP.
AS	Hosting Autonomous System.
Latitude/Longitude	Server geolocation coordinates.

(2) Data Sparsity Emulation. In keeping with existing research Settings [42], to model real-world situations, Table 2 summarizes the dataset partitioning under varying training densities. For both RT (Response Time) and TP (Throughput) tasks, we adopt four levels of data sparsity: 5%, 10%, 15%, and 20% training density, simulating different levels of data availability in real-world scenarios. A fixed 20% of the total dataset is reserved as the validation set for hyperparameter tuning and model calibration, while the remaining samples are assigned to the test set. This split strategy ensures a consistent evaluation framework while systematically analyzing the model's robustness under different degrees of data sparsity.

4.2 Evaluation Metrics

We evaluate the performance of QoSBERT using two widely adopted regression metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). These metrics provide complementary insights into the accuracy and robustness of predicted Quality of Service (QoS) values.

Mean Absolute Error (MAE) measures the average magnitude of absolute differences between the predicted and ground-truth QoS values, and is defined as:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \quad (11)$$

Root Mean Squared Error (RMSE) quantifies the square root of the average squared prediction error, penalizing larger deviations more heavily:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2} \quad (12)$$

TABLE 4: Experimental Results of QoS Prediction Under Multiple Densities on RT Dataset

Methods	Density 5%		Density 10%		Density 15%		Density 20%	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
UPCC	0.698	1.665	0.559	1.466	0.496	1.349	0.464	1.274
LACF	0.631	1.439	0.562	1.338	0.513	1.269	0.477	1.222
PMF	0.623	1.532	0.528	1.329	0.488	1.238	0.469	1.202
NCF	0.472	1.438	0.386	1.314	0.362	1.303	0.352	1.274
LDCF	0.403	1.277	0.364	1.233	0.345	1.169	0.331	1.138
llmQoS	0.409	1.290	0.360	1.224	0.344	1.186	0.327	1.159
FRLN	0.379	1.306	0.344	1.238	0.322	1.201	0.309	1.175
PE-FGL	0.369	1.305	0.339	1.258	0.331	1.233	0.303	1.183
QoSBERT	**0.327**	**1.199**	**0.317**	**1.210**	**0.294**	**1.175**	**0.249**	**1.064**
Gain (%)	**+11.38%**	**+8.12%**	**+6.49%**	**+3.82%**	**+11.18%**	**+4.70%**	**+17.82%**	**+10.06%**

TABLE 5: Experimental Results of QoS Prediction Under Multiple Densities on TP Dataset

Methods	Density 5%		Density 10%		Density 15%		Density 20%	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
UPCC	31.43	77.08	24.70	64.18	22.35	58.95	21.21	56.16
LACF	22.97	55.78	19.44	52.92	17.58	49.56	16.45	47.41
PMF	26.47	67.46	19.83	50.64	16.84	47.48	15.32	43.86
NCF	18.68	54.65	14.40	46.22	13.30	45.35	12.84	44.95
FRLN	14.94	51.62	12.57	44.67	11.37	40.87	11.06	39.60
LDCF	13.84	47.35	12.38	43.48	11.27	39.81	10.84	38.99
llmQoS	13.71	46.65	12.02	42.94	11.15	39.56	10.76	39.36
PE-FGL	13.36	44.71	11.47	41.51	10.35	37.41	9.95	36.35
QoSBERT	**12.61**	**44.31**	**10.75**	**40.54**	**9.58**	**35.65**	**9.14**	**35.93**
Gain (%)	**+5.61%**	**+0.89%**	**+6.28%**	**+2.23%**	**+7.44%**	**+4.70%**	**+8.14%**	**+1.14%**

Here, y_i denotes the ground-truth QoS value (e.g., response time or throughput) for the i-th sample, and $\hat{y}_i$ is the predicted mean μ_i of the Gaussian output from the model. N is the total number of test records.

Lower MAE and RMSE values indicate higher prediction accuracy. In our setting, both metrics are computed over the predicted means from the Monte Carlo ensemble, thereby capturing both point-wise precision and the benefits of uncertainty-aware inference.

4.3 Baseline Methods

We compare our QoSBERT approach with the following methods:

UPCC [27]: A classic collaborative filtering method that predicts QoS values based on similarity between users with shared service histories.

LACF [14]: A location-aware collaborative filtering approach that incorporates both user and service locations into similarity computation to improve QoS prediction accuracy.

PMF [51]: A scalable matrix factorization method that models user-service interactions probabilistically, suitable for large-scale and sparse QoS datasets.

NCF [52]: A neural collaborative filtering model that replaces matrix factorization with a multi-layer perceptron to learn complex user-service interactions.

LDCF [18]: A deep learning–based CF model that incorporates location embeddings and a similarity correction mechanism to improve QoS prediction under sparse data.

FRLN [41]: A federated learning framework using residual ladder networks to extract deep features while preserving user privacy during collaborative QoS prediction.

llmQoS [43]: This approach transforms user and server-side attributes into descriptive text and fine-tunes a large language model.

PE-FGL [42]: A privacy-enhanced federated graph learning model that expands local invocation graphs and securely aggregates learned parameters for accurate and private QoS prediction.

5 EXPERIMENTS

In this section, we conduct thorough experiments to evaluate the prediction performance of QoSBERT against state-of-the-art baselines. We also delve into the influence of various modules and parameters on QoSBERT's performance through ablation studies. Experiments were performed 3 times and the average is reported.

5.1 Prediction Performance Comparison

Table 4 reports the prediction results for response time (RT) under varying training densities. Across all baselines, QoSBERT achieves the lowest MAE and RMSE values in each density setting, demonstrating consistent superiority. Compared to the strongest baseline PE-FGL, QoSBERT reduces MAE by 11.38% and RMSE by 8.12% at 5% density. As the data density increases, the performance gains further expand, reaching up to 17.82% MAE and 10.06% RMSE improvements at 20% density. These results highlight QoSBERT's robustness and accuracy in low-resource and high-sparsity scenarios for RT prediction.

Table 5 presents the performance comparison on throughput (TP). Similarly, QoSBERT consistently outperforms all baselines across training densities, with noticeable gains over PE-FGL. Specifically, QoSBERT improves MAE by 5.61% and RMSE by 0.89% at 5% density, and maintains strong performance with improvements of 8.14% in MAE and 1.14% in RMSE at 20% density. The performance advantage is particularly evident in sparse data conditions, underscoring the model's ability to generalize well across different QoS metrics.

In summary, the results demonstrate that QoSBERT significantly improves QoS prediction accuracy across both response time and throughput tasks. Its semantic-rich feature encoding, multi-layer pooling mechanism, and uncertainty-aware regression jointly contribute to robust performance under diverse sparsity settings, making it highly competitive among state-of-the-art approaches.

5.2 QoSBERT VS Supervised Fine-Tuning (SFT)

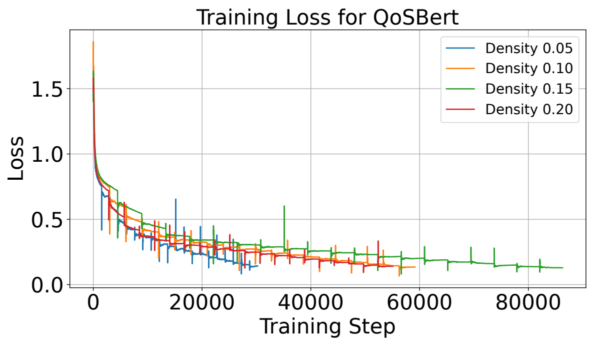
To further validate the architectural advantages of QoSBERT, we compare it against a widely-adopted supervised fine-tuning (SFT) baseline. This SFT paradigm follows the standard practice used in numerous PLM-based downstream tasks—including text classification and code understanding benchmarks such as CodeXGLUE [53]—where the final-layer `[CLS]` token is directly fed into a lightweight MLP regression head. This design is simple and effective in many settings, but lacks both representation enhancement and uncertainty modeling.

In contrast, QoSBERT introduces two core innovations:

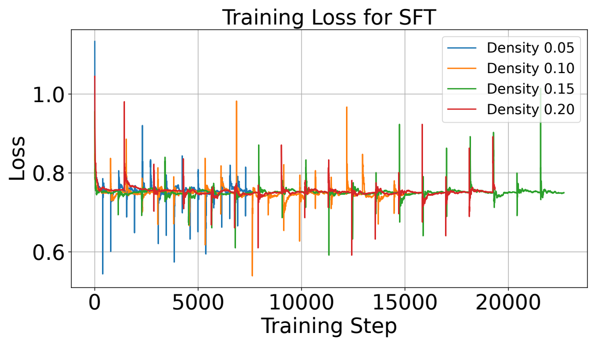
(1) Representation aggregation: Unlike SFT, which exclusively relies on the last layer's `[CLS]` token, QoSBERT fuses semantic features across the top-K layers using mean, max, and attention-based pooling to enrich contextual representation. (2) Uncertainty-aware training: QoSBERT incorporates Monte Carlo Dropout (MC-Dropout) during both training and inference, enabling better regularization and calibrated predictions under sparse and noisy feature conditions.

Fig. 2 shows the training loss curves of both models across four data densities. Notably, the standard SFT baseline exhibits almost no effective convergence even after 20 epochs. At all density, its loss remains stagnant and significantly higher than QoSBERT's. This failure to fit is likely due to the combination of sparse input features and insufficient inductive bias, which prevents the shallow SFT architecture from learning meaningful correlations in low-resource settings. Quantitatively, the standard SFT baseline (i.e., CLS + MLP) yields an average MAE around **0.74** and RMSE around **2.10** across all four density settings, which is substantially worse than QoSBERT's corresponding results. This confirms that a shallow fine-tuning approach cannot adequately capture the complex and sparse semantics present in real-world service descriptions.

In contrast, QoSBERT demonstrates faster convergence and consistently achieves lower loss values across all densities. These results highlight that architectural innovations—such as richer feature aggregation and uncertainty-aware learning—are crucial for training robust and effective QoS predictors under limited data availability.

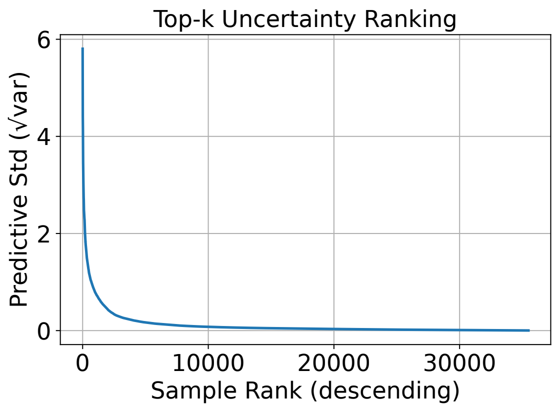


(a) QoSBERT Training Loss



(b) SFT Training Loss

Fig. 2: Training loss curves of QoSBERT and standard SFT under various data densities



(a) Top-K Uncertainty Ranking

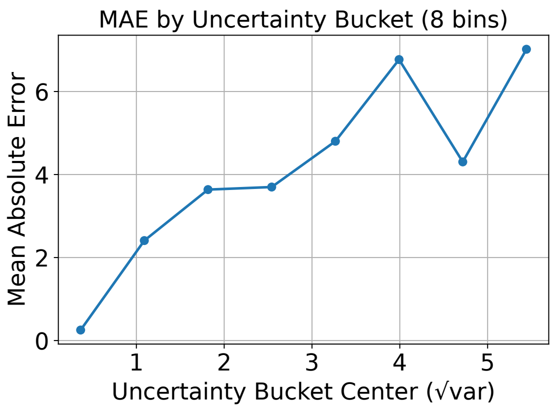
Fig. 3: Top-K uncertainty curve showing the ranked predictive standard deviation $\sqrt{\text{Var}}$ of all samples.

5.3 Uncertainty Analysis and Visualization

To better understand the predictive uncertainty of our model and its reliability in the QoS prediction task, we conduct a thorough analysis using four complementary visualizations. These visualizations reveal the behavior of uncertainty estimates and validate their calibration quality, supporting the practical utility of our approach in risk-sensitive applications.

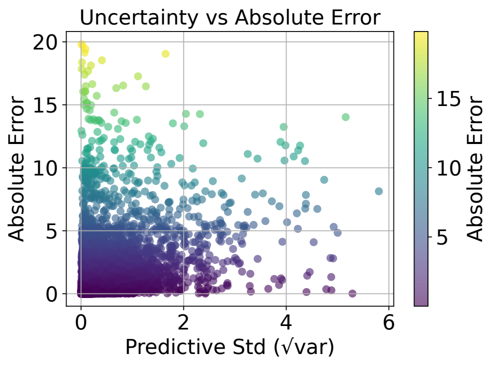
(1) Top-K Uncertainty Curve

In Fig. 3, we sort all samples in descending order of the predicted standard deviation $\sqrt{\text{Var}}$ and visualize the ranked curve to assess the distribution of model uncertainty. The resulting curve



(a) MAE by Uncertainty Bucket (8 bins)

Fig. 4: Relationship between predicted uncertainty and MAE across 8 uncertainty buckets.



(a) Uncertainty vs. Absolute Error Scatter

Fig. 5: Scatter plot of predicted standard deviation $\sqrt{\text{Var}}$ versus absolute error $|\hat{y}-y|$, with color indicating error magnitude.

demonstrates a smooth and steep decline at the head, followed by a long tail of low-uncertainty samples. Specifically, the top 5% of predictions exhibit significantly higher uncertainty—exceeding 0.25 in standard deviation—while over 80% of the samples fall below 0.1, indicating the model is confident on the majority of inputs.

This pattern suggests that the model's uncertainty estimation is highly structured: it can effectively distinguish between ambiguous and well-understood inputs. For instance, high-uncertainty samples typically correspond to rare or noisy service requests, such as API invocations with missing parameters, uncommon QoS patterns, or short input sequences lacking contextual signals. Conversely, low-uncertainty predictions are often associated with frequently seen service-user pairs or structurally regular code snippets.

This stratification is critical for practical deployment. In service-oriented systems, top-ranked uncertain samples can be selectively routed through fallback procedures—such as ensemble voting, human-in-the-loop validation, or re-querying alternative services. As a result, the system can avoid overconfident but incorrect decisions and improve its overall reliability. The presence of a clear uncertainty head, as shown in the plot, ensures that such risk-aware mechanisms can be applied with clear thresholds.

(2) Uncertainty Bucket vs. MAE

Fig. 4 illustrates the relationship between predicted uncertainty and actual prediction error by grouping samples into 8 equally spaced buckets based on their predicted standard deviation $\sqrt{\text{Var}}$. For each bucket, we compute the average mean absolute error (MAE), revealing how error scales with uncertainty.

A clear positive correlation emerges: the lowest-uncertainty bucket (mean $\sqrt{\text{Var}} < 0.05$) achieves an average MAE below 0.35, while the highest-uncertainty bucket (mean $\sqrt{\text{Var}} > 0.25$) exhibits an MAE exceeding 1.2. This monotonic increase confirms that the model's uncertainty signal is calibrated in the relative sense—predictions flagged as uncertain are empirically more error-prone.

This result has strong implications for uncertainty-aware QoS prediction. For instance, in latency-sensitive service orchestration, the system can automatically lower trust in predictions falling into higher-uncertainty bins, or even defer execution if the associated MAE risk exceeds a pre-defined operational threshold (e.g., 1.0 ms absolute deviation). Additionally, operators can prioritize model retraining on examples drawn from high-uncertainty buckets to reduce epistemic risk and improve performance in underrepresented regions of the input space.

(3) Uncertainty vs. Absolute Error Scatter

Fig. 5 provides a fine-grained view of the relationship between model-estimated uncertainty and actual prediction error by plotting each sample as a point in the $(\sqrt{\text{Var}}, |\hat{y}-y|)$ space. We additionally color-coded the points based on error magnitude to highlight risk concentration patterns.

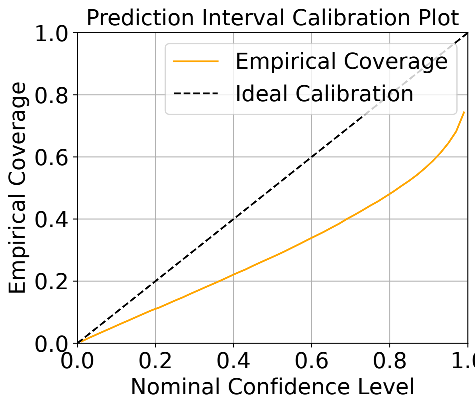
Overall, we observe a positive correlation between uncertainty and error—samples with higher predictive standard deviation tend to exhibit larger deviations from ground truth. For instance, a dense cluster of points with $\sqrt{\text{Var}} < 0.1$ shows absolute errors mostly below 0.4, while samples with $\sqrt{\text{Var}} > 0.3$ show a much wider error distribution, including many with error $>$ 1.5. However, some high-error samples still exhibit low predicted uncertainty, suggesting residual aleatoric uncertainty or blind spots in model calibration.

This scatter pattern indicates that while uncertainty is not a perfect proxy for error, it statistically identifies high-risk predictions. For example, by thresholding uncertainty at $\sqrt{\text{Var}} > 0.25$, one can isolate a subset of predictions where the probability of error exceeding 1.0 rises significantly. In production QoS prediction systems, such filtering can be used to trigger redundancy (e.g., replicate service invocation), fallback logic, or additional data collection.

(4) Calibration Curve (Coverage Plot)

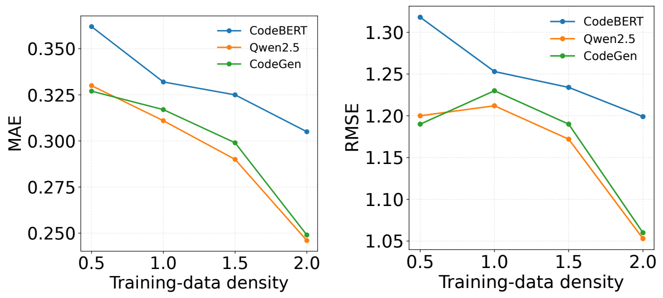
To evaluate the calibration quality of the uncertainty estimates, Fig. 6 plots the empirical coverage probabilities of predicted confidence intervals against the expected confidence levels (e.g., 90% confidence should ideally contain the true value 90% of the time). Each point on the curve represents a fixed confidence level $\alpha \in (0.05, 0.99)$, with the y-axis showing the fraction of test points for which the true label lies within the corresponding predictive interval $[\mu - z_\alpha \cdot \sqrt{\text{Var}}, \mu + z_\alpha \cdot \sqrt{\text{Var}}]$.

A perfectly calibrated model would yield a diagonal line $y = x$. Our observed curve generally tracks the diagonal but exhibits under-confidence in the middle range: for instance, the 80% confidence interval actually contains the true value approximately 86% of the time. This suggests that the model tends to overestimate its uncertainty slightly—producing intervals that are too wide for a given nominal confidence level. At high confidence levels (e.g.,



(a) Prediction Interval Calibration Plot

Fig. 6: Calibration curve comparing empirical coverage of predictive intervals to their nominal confidence levels.



(a) MAE (b) RMSE

Fig. 7: Effect of the different pretrained models

95%-99%), the coverage closely matches the ideal line, confirming reliability in the tail.

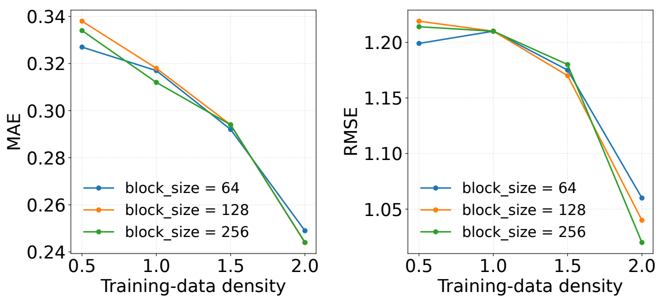
From a practical perspective, well-calibrated uncertainty allows one to quantify trustworthiness per prediction. In service composition or selection tasks, QoS estimates with narrow confidence intervals can be acted upon more assertively, whereas those with wide or ill-calibrated intervals should be treated conservatively. The coverage plot thus validates that our uncertainty estimates are not only statistically meaningful but also operationally actionable in downstream QoS-sensitive decision flows.

5.4 The impact of different pre-trained models

In this section, we evaluate the influence of different pre-trained language models on the performance of service quality prediction. We compare three representative models with similar parameter scales (approximately 500 million parameters): CodeBERT [54], Qwen2.5-0.5B [22], and CodeGen [46]. The results across multiple data densities are shown in Fig. 7.

These models are selected to represent different pretraining paradigms:

- **CodeBERT** is a code-specific encoder model trained predominantly on source code and associated comments, making it a strong representative of *code-centric* representations.
- **CodeGen** follows a multi-stage training procedure, where early stages are pre-trained on general natural language corpora (e.g., Wikipedia, academic text), followed by large-scale code pretraining. It thus reflects a *hybrid code + natural language* model.



(a) MAE (b) RMSE

Fig. 8: Effect of the Transformer block size (64, 128, 256)

- **Qwen2.5-0.5B-Instruct** is a general-purpose, instruction-tuned large language model trained extensively on diverse natural language tasks. It serves as a representative of *general-domain* LLMs.

All three models fall within the 0.5B parameter range, striking a practical balance between capacity and inference efficiency for service recommendation systems.

From Fig. 7(a), we observe that Qwen achieves the lowest MAE across most data densities, especially at high densities where it outperforms CodeBERT by up to 19.3% (D1.4). This suggests that Qwen may be better adapted to capturing fine-grained user-service interactions due to its advanced pre-training objectives. CodeGen also performs competitively, especially under sparse data (e.g., MAE = 0.327 at D1.1), highlighting its robustness under limited supervision.

RMSE results in Fig. 7(b) further support these findings: both Qwen and CodeGen consistently outperform CodeBERT, with the largest improvement of 20.1% in RMSE occurring at D1.4 for Qwen. These improvements demonstrate that model choice plays a crucial role in reducing prediction variance, particularly in federated or decentralized settings with high data heterogeneity.

Overall, these results confirm the importance of selecting suitable pre-trained models under computational constraints. Models like Qwen and CodeGen offer a favorable trade-off between prediction accuracy and system efficiency, and thus serve as strong backbones for our QoSBERT.

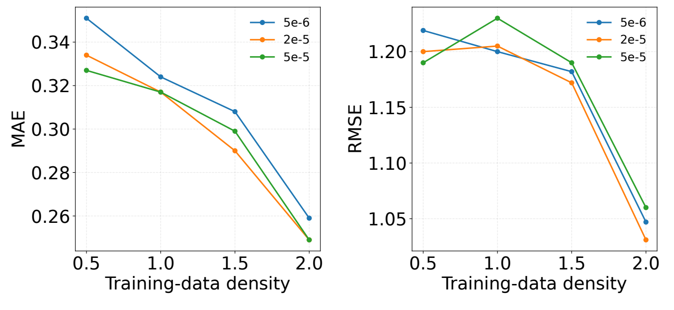
5.5 Ablation experiment

In this section, we conduct ablation experiments to evaluate the effectiveness of our proposed methods. The experiments are divided into three parts: 1) Impact of block size on Prediction Accuracy, and 2) Impact of Learning rate on Prediction Accuracy, and 3) Impact of τ on Prediction Accuracy and 4) Impact of Pooling Strategies on QoS Prediction.

5.5.1 Impact of block size on prediction accuracy

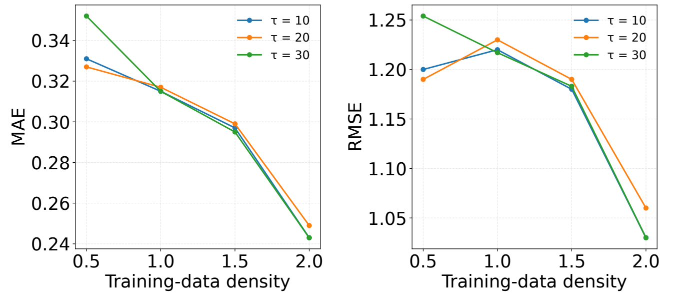
Fig.8 reports the mean absolute error (MAE) and root-mean-squared error (RMSE) obtained by QoSBERT when the maximum input token length In the original implementation we reuse the block size argument of the HuggingFace Trainer to control the sliding-window length during sequence truncation. varies among 64, 128, and 256. The four points on each curve correspond to the progressively denser training subsets D1.1(0.05), D1.2 (0.10), D1.3 (0.15) and D1.4 (0.20).

With the very restricted feature vocabulary currently available, enlarging the receptive field from the default 32 to 64 tokens immediately translates into a substantial performance jump: on



(a) MAE (b) RMSE

Fig. 9: Effect of learning rate on model performance across different training densities



(a) MAE (b) RMSE

Fig. 10: Effect of temperature on model performance

D1.1 the MAE falls by **21.5%** (from 0.416 to 0.327) while RMSE improves by **15.9%**. This confirms that the extra contextual tokens carry complementary semantics that the model could not capture with smaller windows.

Extending the window further to 128 tokens produces a marginal yet consistent improvement—typically under 3% for both metrics—while the step from 128 to 256 becomes practically negligible. We attribute this plateau to two factors: (i) the service/user descriptions we can harvest at present seldom exceed 120 tokens after templating, hence most sequences are already fully covered with block size= 128; (ii) the regression head, which aggregates information by mean–max–attention pooling, is robust to minor truncation of the tail tokens. Despite the limited benefits, using a larger block size does *not* hurt performance. The curves in Fig. 8 remain virtually flat between 128 and 256 across all data densities, indicating that the extra tokens neither introduce noise nor destabilise optimisation.

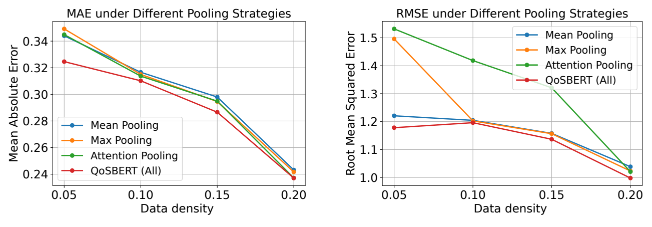
Given the present feature coverage and the trade-off between memory footprint and accuracy, we set block size $= 64$ as the default for all subsequent experiments. This value captures almost all informative tokens while keeping GPU utilisation modest, and it leaves headroom for future feature engineering without requiring changes to the architecture.

5.5.2 Impact of Learning rate on prediction accuracy

As shown in Fig. 9, a learning rate of $\lambda = 2 \times 10^{-5}$ achieves the best overall performance across data densities, consistently reducing both MAE and RMSE compared to lower and higher settings. This indicates a good balance between convergence stability and gradient responsiveness, and is therefore adopted as the default value in all remaining experiments.

5.5.3 Impact of τ on prediction accuracy

As illustrated in Fig. 10, varying the temperature scaling parameter τ yields notable differences in prediction accuracy. Overall,



(a) MAE (b) RMSE

Fig. 11: Effect of pooling strategies on prediction accuracy

$\tau = 20$ achieves the best performance, minimizing both MAE and RMSE across multiple data densities. A lower value (e.g., $\tau = 10$) tends to under-correct model uncertainty, while a higher value (e.g., $\tau = 30$) can over-amplify prediction variance, leading to suboptimal performance. These results demonstrate that appropriate temperature calibration is crucial for uncertainty-aware QoS estimation and highlight $\tau = 20$ as a well-balanced choice adopted in our final model configuration.

5.5.4 Impact of Pooling Strategies on QoS Prediction

To further investigate the effectiveness of our pooling design, we conduct an ablation study comparing different token aggregation strategies. Specifically, we examine the impact of using single pooling operations—mean, max, or attention pooling—versus the multi-pooling fusion adopted in QoSBERT. This analysis helps to clarify whether the performance gains stem from the pooling architecture or other components.

As illustrated in Fig. 11, different pooling strategies significantly affect the regression performance. Mean pooling captures the overall semantic distribution, while max pooling emphasizes salient tokens. Attention pooling adaptively reweights token importance based on task signals. Across varying dropout rates, all three single-strategy variants yield competitive results. However, the multi-pooling fusion adopted in QoSBERT consistently outperforms individual variants on both MAE and RMSE. This highlights the complementarity of token-level summary mechanisms, especially in sparsely labeled, noisy QoS scenarios.

To isolate the effect of the pooling strategies, we conduct an ablation study comparing our default *multi-pooling* fusion (mean $\oplus$ max $\oplus$ attention) with an extended variant that adds *stochastic* and *K-max* pooling, yielding a five-way configuration.

TABLE 6: Ablation on pooling strategies under varying training densities.

Train Density	3-Pooling (mean+max+attn)		5-Pooling (extended)	
	MAE ↓	RMSE ↓	MAE ↓	RMSE ↓
0.05	0.3246	1.1782	0.3294	1.1855
0.10	0.3102	1.1961	0.3122	1.1902
0.15	0.2866	1.1367	0.2980	1.1690
0.20	0.2371	0.9978	0.2449	1.0655

As shown in Table 6, expanding the pooling set to five operators yields accuracy that is comparable but not consistently superior to the default fusion. In low-resource regimes (train densities 0.05 and 0.10), the 3-pooling variant attains lower MAE and competitive or lower RMSE. At higher densities (0.15 and 0.20), the two configurations achieve similar accuracy with small trade-offs across metrics.

Additional pooling increases architectural complexity and can reduce interpretability without delivering consistent gains. We therefore retain the 3-pooling fusion as the default in QoSBERT for its balance of accuracy, stability, and efficiency. The framework remains modular, so alternative pooling modules can be substituted for domain-specific needs.

6 CONCLUSION

In this paper, we propose **QoSBERT**, a novel uncertainty-aware framework for service quality prediction that leverages the representation power of pre-trained language models. Unlike traditional QoS prediction methods that rely solely on numerical or structural features, QoSBERT models textual service descriptions through a Transformer-based encoder, enabling semantic-aware learning. Comprehensive experiments on the widely-used WS-DREAM dataset demonstrate that QoSBERT outperforms several strong baselines in both MAE and RMSE metrics. Beyond accuracy, we systematically assess the quality of uncertainty estimation through top-K ranking plots, error-based binning, and calibration curves. These analyses confirm that the predicted variances are well-structured and informative—higher uncertainties are generally associated with larger prediction errors, and the model exhibits moderate under-confidence that can be mitigated by post-hoc calibration.

Our work bridges the gap between accuracy and trustworthiness in QoS prediction, laying a foundation for uncertainty-aware service recommendation and selection. In the future, we plan to explore ensemble-based calibration strategies and integrate uncertainty into downstream tasks such as QoS-aware reranking and risk-aware orchestration.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers and the editorial team for their constructive feedback. This work was supported by the College of Big Data and Software, Chongqing University.

REFERENCES

[1] H. S. M. Muslim and R. et al., "S-rap: relevance-aware qos prediction in web-services and user contexts," *Knowledge and Information Systems*, vol. 64, no. 7, pp. 1997–2022, 2022.

[2] Z. Zheng, X. Li, M. Tang, F. Xie, and M. R. Lyu, "Web service qos prediction via collaborative filtering: A survey," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2455–2472, 2020.

[3] Z. Wang, S. Zhu, J. Li, W. Jiang, K. Ramakrishnan, M. Yan, X. Zhang, and A. X. Liu, "Deepscaling: Autoscaling microservices with stable cpu utilization for large scale production cloud systems," *IEEE/ACM Transactions on Networking*, 2024.

[4] J. Liu and Y. Chen, "A personalized clustering-based and reliable trust-aware qos prediction approach for cloud service recommendation in cloud manufacturing," *Knowledge-Based Systems*, vol. 174, pp. 43–56, 2019.

[5] L. Yao, Q. Z. Sheng, A. H. Ngu, J. Yu, and A. Segev, "Unified collaborative and content-based web service recommendation," *IEEE Transactions on Services Computing*, vol. 8, no. 3, pp. 453–466, 2014.

[6] S. K. Gavvala, C. Jatoth, G. Gangadharan, and R. Buyya, "Qos-aware cloud service composition using eagle strategy," *Future Generation Computer Systems*, vol. 90, pp. 273–290, 2019.

[7] S. Sefati and N. J. Navimipour, "A qos-aware service composition mechanism in the internet of things using a hidden-markov-model-based optimization algorithm," *IEEE Internet of Things Journal*, vol. 8, no. 20, pp. 15 620–15 627, 2021.

[8] J. Park, B. Choi, C. Lee, and D. Han, "Graf: a graph neural network based proactive resource allocation framework for slo-oriented microservices," in *International Conference on emerging Networking EXperiments and Technologies*, 2021, pp. 154–167.

[9] W. Hussain, J. M. Merigó, M. R. Raza, and H. Gao, "A new qos prediction model using hybrid iowa-anfis with fuzzy c-means, subtractive clustering and grid partitioning," *Information Sciences*, vol. 584, pp. 280–300, 2022.

[10] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, "Personalized qos prediction forweb services via collaborative filtering," in *Ieee international conference on web services (icws 2007)*. IEEE, 2007, pp. 439–446.

[11] Z. Liu, Q. Z. Sheng, X. Xu, D. Chu, and W. E. Zhang, "Context-aware and adaptive qos prediction for mobile edge computing services," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 400–413, 2019.

[12] S. H. Ghafouri, S. M. Hashemi, and P. C. Hung, "A survey on web service qos prediction methods," *IEEE Transactions on Services Computing*, vol. 15, no. 4, pp. 2439–2454, 2020.

[13] G. Zou, M. Jiang, S. Niu, H. Wu, S. Pang, and Y. Gan, "Qos-aware web service recommendation with reinforced collaborative filtering," in *Service-Oriented Computing: 16th International Conference, ICSOC 2018, Hangzhou, China, November 12-15, 2018, Proceedings 16*. Springer, 2018, pp. 430–445.

[14] M. Tang, Y. Jiang, J. Liu, and X. Liu, "Location-aware collaborative filtering for qos-based service recommendation," in *2012 IEEE 19th international conference on web services*. IEEE, 2012, pp. 202–209.

[15] Z. Chen, L. Shen, and F. Li, "Exploiting web service geographical neighborhood for collaborative qos prediction," *Future Generation Computer Systems*, vol. 68, pp. 248–259, 2017.

[16] G. Zou, S. Wu, S. Hu, C. Cao, Y. Gan, B. Zhang, and Y. Chen, "Ncrl: Neighborhood-based collaborative residual learning for adaptive qos prediction," *IEEE Transactions on Services Computing*, 2022.

[17] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware qos prediction with neural collaborative filtering for internet-of-things services," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4532–4542, 2019.

[18] Y. Zhang, C. Yin, Q. Wu, Q. He, and H. Zhu, "Location-aware deep collaborative filtering for service recommendation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 6, pp. 3796–3807, 2019.

[19] T. Lu, X. Zhang, Z. Wang, and M. Yan, "A feature distribution smoothing network based on gaussian distribution for qos prediction," in *2023 IEEE International Conference on Web Services (ICWS)*. IEEE Computer Society, 2023, pp. 687–694.

[20] Z. Wang, X. Zhang, M. Yan, L. Xu, and D. Yang, "Hsa-net: Hidden-state-aware networks for high-precision qos prediction," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1421–1435, 2021.

[21] F. Ye, Z. Lin, C. Chen, Z. Zheng, and H. Huang, "Outlier-resilient web service qos prediction," in *Proceedings of the Web Conference 2021*, 2021, pp. 3099–3110.

[22] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei *et al.*, "Qwen2. 5 technical report," *arXiv preprint arXiv:2412.15115*, 2024.

[23] K. Lee, J. Park, and J. Baik, "Location-based web service qos prediction via preference propagation for improving cold start problem," in *IEEE International Conference on Web Services*. IEEE, 2015, pp. 177–184.

[24] X. Chen, X. Liu, Z. Huang, and H. Sun, "Regionknn: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation," in *IEEE international conference on web services*. IEEE, 2010, pp. 9–16.

[25] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Qos-aware web service recommendation by collaborative filtering," *IEEE Transactions on services computing*, vol. 4, no. 2, pp. 140–152, 2010.

[26] Z. Chen, L. Shen, F. Li, and D. You, "Your neighbors alleviate cold-start: On geographical neighborhood influence to collaborative web service qos prediction," *Knowledge-Based Systems*, vol. 138, pp. 188–201, 2017.

[27] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[28] Z. Tan and L. He, "An efficient similarity measure for user-based collaborative filtering recommender systems inspired by the physical resonance principle," *IEEE Access*, vol. 5, pp. 27 211–27 228, 2017.

[29] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 39–46.

[30] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q. Zhu, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 3, pp. 579–592, 2015.

[31] Y. Shi, M. Larson, and A. Hanjalic, "Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges," *ACM Computing Surveys (CSUR)*, vol. 47, no. 1, pp. 1–45, 2014.

[32] Y. Zhang, Z. Zheng, and M. R. Lyu, "Wspred: A time-aware personalized qos prediction framework for web services," in *IEEE International Symposium on Software Reliability Engineering*. IEEE, 2011, pp. 210–219.

[33] Z. Luo, L. Liu, and Y. et al., "Latent ability model: A generative probabilistic learning framework for workforce analytics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 923–937, 2018.

[34] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[35] Q. Zhou, H. Wu, K. Yue, and C.-H. Hsu, "Spatio-temporal context-aware collaborative qos prediction," *Future Generation Computer Systems*, vol. 100, pp. 46–57, 2019.

[36] H. Wang, L. Wang, Q. Yu, Z. Zheng, A. Bouguettaya, and M. R. Lyu, "Online reliability prediction via motifs-based dynamic bayesian networks for service-oriented systems," *IEEE Transactions on Software Engineering*, vol. 43, no. 6, pp. 556–579, 2016.

[37] R. Xiong, J. Wang, N. Zhang, and Y. Ma, "Deep hybrid collaborative filtering for web service recommendation," *Expert systems with Applications*, vol. 110, pp. 191–205, 2018.

[38] M. Liu, H. Xu, Q. Z. Sheng, and Z. Wang, "Qosgnn: Boosting qos prediction performance with graph neural networks," *IEEE Transactions on Services Computing*, 2023.

[39] W. Zhang, L. Xu, M. Yan, Z. Wang, and C. Fu, "A probability distribution and location-aware resnet approach for qos prediction," *Journal of Web Engineering*, vol. 20, no. 4, pp. 1251–1290, 2021.

[40] Z. Wang, X. Zhang, M. Yan, L. Xu, and D. Yang, "Hsa-net: Hidden-state-aware networks for high-precision qos prediction," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1421–1435, 2021.

[41] G. Zou, W. Yu, S. Hu, Y. Gan, B. Zhang, and Y. Chen, "Frln: Federated residual ladder network for data-protected qos prediction," *IEEE Transactions on Services Computing*, 2024.

[42] G. Zou, Z. Yan, S. Hu, Y. Gan, B. Zhang, and Y. Chen, "Privacy-enhanced federated expanded graph learning for secure qos prediction," *IEEE Transactions on Services Computing*, 2025.

[43] H. Liu, Z. Zhang, H. Li, Q. Wu, and Y. Zhang, "Large language model aided qos prediction for service recommendation," *arXiv preprint arXiv:2408.02223*, 2024.

[44] Y. Akhauri, B. Lewandowski, C.-H. Lin, A. N. Reyes, G. C. Forbes, A. Wongpanich, B. Yang, M. S. Abdelfattah, S. Perel, and X. Song, "Performance prediction for large systems via text-to-text regression," *arXiv preprint arXiv:2506.21718*, 2025.

[45] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[46] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong, "Codegen: An open large language model for code with multi-turn program synthesis," *arXiv preprint arXiv:2203.13474*, 2022.

[47] W. De Vries, A. Van Cranenburgh, and M. Nissim, "What's so special about bert's layers? a closer look at the nlp pipeline in monolingual and multilingual models," *arXiv preprint arXiv:2004.06499*, 2020.

[48] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[49] H. Wu, K. Xu, L. Song, L. Jin, H. Zhang, and L. Song, "Domain-adaptive pretraining methods for dialogue understanding," *arXiv preprint arXiv:2105.13665*, 2021.

[50] A. Yang, B. Yang, B. Hui, B. Zheng, B. Yu, C. Zhou, C. Li, C. Li, D. Liu, F. Huang, G. Dong, H. Wei, H. Lin, J. Tang, J. Wang, J. Yang, J. Tu, J. Zhang, J. Ma, J. Xu, J. Zhou, J. Bai, J. He, J. Lin, K. Dang, K. Lu, K. Chen, K. Yang, M. Li, M. Xue, N. Ni, P. Zhang, P. Wang, R. Peng, R. Men, R. Gao, R. Lin, S. Wang, S. Bai, S. Tan, T. Zhu, T. Li, T. Liu, W. Ge, X. Deng, X. Zhou, X. Ren, X. Zhang, X. Wei, X. Ren, Y. Fan, Y. Yao, Y. Zhang, Y. Wan, Y. Chu, Y. Liu, Z. Cui, Z. Zhang, and Z. Fan, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024.

[51] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," *Advances in neural information processing systems*, vol. 20, 2007.

[52] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[53] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang *et al.*, "Codexglue: A machine learning benchmark dataset for code understanding and generation," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

[54] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang *et al.*, "Codebert: A pre-trained model for programming and natural languages," *arXiv preprint arXiv:2002.08155*, 2020.