QoSBERT: A 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 present QoSBERT, a novel end-to-end framework that leverages pre-trained language models to automatically encode rich, natural-language descriptions of services and integrates a Monte Carlo Dropout–based uncertainty estimation mechanism to quantify predictive confidence. QoSBERT applies attentive pooling over contextualized embeddings and a lightweight multi-layer perceptron regressor, fine-tuned jointly to minimize absolute error. We further exploit the resulting uncertainty estimates to select high-quality training samples, improving robustness in low-resource settings. 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, Latent state, Mixture of experts

arXiv:2505.07863v1 [cs.CL] 9 May 2025

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 or sparse access records. For example, zhang et al. encoded the user location information into an integer value to encode the user location information 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

- *Ziliang Wang is with Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education; School of Computer Science, Peking University, Beijing, 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 in the Department of Computer Science at the University of Victoria, Canada
- *Ziliang Wang is the corresponding authors.*
  E-mail: wangziliang@pku.edu.cn

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 natural language descriptions and encodes them using powerful pre-trained models (e.g., Qwen2.5 [22]). This enables the model to capture rich semantic features embedded in service texts and user contexts, allowing for better generalization across different domains, even with limited numerical inputs. This has the intuitive benefit that features that were previously difficult to model numerically can be exploited.

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, the first framework that reformulates QoS prediction as a semantic regression task using pre-trained language models, allowing effective encoding of descriptive user-service information without requiring explicit numerical feature engineering.

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.

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-Based QoS prediction Approaches. 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 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.

3 Approach

The architecture of QoSBERT, as illustrated in Fig. 2, 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

3.1.1 Semantic Feature Encoding from Service and User Metadata

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 semantic feature construction strategy that transforms structured metadata (e.g., IP address, AS number, country, region) into unified natural language descriptions. Specifically, we parse the raw metadata embedded in each QoS record and generate a standardized textual summary using predefined templates.

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 [43], CodeGen [44]), 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.

Algorithm 1 QoSBERT: QoS Prediction with Uncertainty Estimation

Require: QoS records $L = \{(U_i, S_i, T_i)\}_{i=1}^{N}$; dropout iterations T; temperature parameter τ

Ensure: Predicted QoS μ and calibrated variance σ_{cal}^2

// Feature Construction
for $i = 1$ to N **do**
  Generate description $x_i = \texttt{desc}(U_i) \| \texttt{desc}(S_i)$
end for
// Contextual Encoding
for $i = 1$ to N **do**
  Tokenize and encode x_i with pretrained Transformer to get $H_i^{(1)}, ..., H_i^{(L)}$
  Fuse top-K layers: $H_i^{\text{fused}} = \frac{1}{K} \sum_{l=L-K+1}^{L} H_i^{(l)}$
  Apply mean, max, and attention pooling:

$$v_i = \text{Mean}(H_i^{\text{fused}}) + \text{Max}(H_i^{\text{fused}}) + \text{Attn}(H_i^{\text{fused}})$$

end for
// Monte Carlo Dropout Sampling
for $t = 1$ to T **do**
  Apply dropout and get $(\mu_i^{(t)}, \sigma_i^{2(t)}) = f_\theta^{(t)}(v_i)$
end for
Compute aggregated prediction:

$$\mu_i = \frac{1}{T} \sum_{t=1}^{T} \mu_i^{(t)}, \quad \sigma_i^2 = \frac{1}{T} \sum_{t=1}^{T} \sigma_i^{2(t)}$$

// Temperature Scaling (Inference Only)

$$\sigma_{i,\text{cal}}^2 = \exp(2 \cdot \log \tau) \cdot \sigma_i^2$$

return μ_i, $\sigma_{i,\text{cal}}^2$

3.2 Fusion Modeling with Pre-trained Models

3.2.1 Contextual Representation

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:

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

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:

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

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

Multi-Layer Feature Fusion. Recent empirical studies [43], [45] 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,

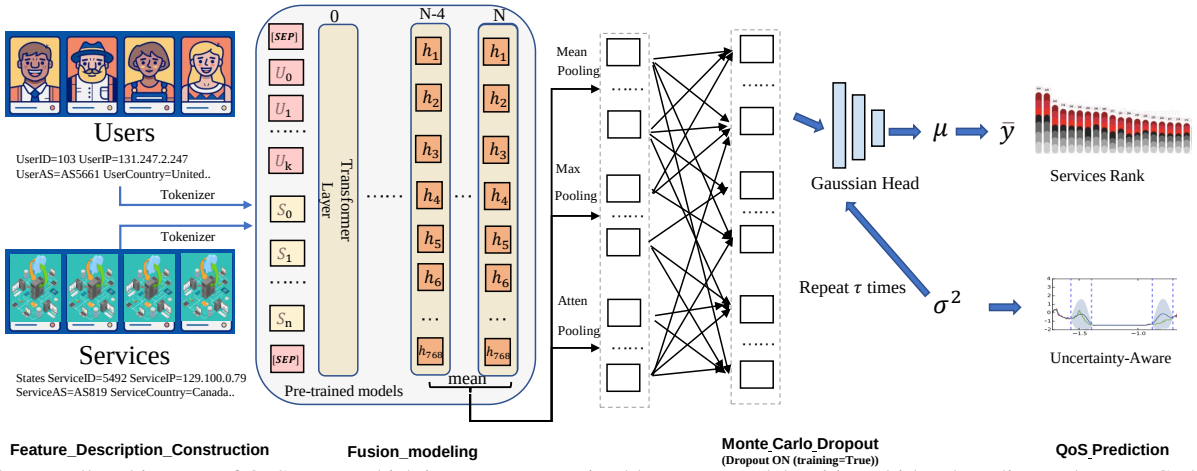


Fig. 1: The overall architecture of QoSBERT, which integrates pre-trained language models with multi-head pooling and Monte Carlo dropout to enable uncertainty-aware QoS prediction based on user-service textual descriptions.

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:

- **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:

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

This vector $\mathbf{v}_{\text{seq}}$ 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 probabilistic regression strategy based on Monte Carlo Dropout (MC Dropout) [**?**]. During inference, we perform T stochastic forward passes through the model with dropout layers activated, and collect multiple samples of the predicted Gaussian parameters $(\mu^{(i)}, \sigma^{2(i)})$.

Formally, for each pass $i = 1, ..., T$, the regression head outputs:

$$(\mu^{(i)}, \sigma^{2(i)}) = \mathscr{F}_{\theta}^{(i)}(\mathbf{v}_{\text{seq}})$$

The final prediction is obtained by averaging the sampled means and variances:

$$\mu = \frac{1}{T} \sum_{i=1}^{T} \mu^{(i)}, \quad \sigma^2 = \frac{1}{T} \sum_{i=1}^{T} \sigma^{2(i)}$$

This sampling procedure allows us to capture the epistemic uncertainty associated with the model's predictions.

3.3.2 Temperature Scaling for Calibration

To improve the calibration of the predicted uncertainty, we introduce a learnable temperature parameter τ, which scales the estimated variance:

$$\sigma_{\text{cal}}^2 = \tau^2 \cdot \sigma^2, \quad \tau = \exp(\log \tau)$$

The temperature parameter $\log \tau$ is optimized on a held-out calibration set by minimizing the Gaussian negative log-likelihood (NLL):

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

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:

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

Here, λ is a hyperparameter that controls the trade-off between calibration and prediction precision. At inference time, we report both the predicted mean μ and calibrated uncertainty σ_{cal}^2 to support risk-aware decision making in QoS-sensitive applications.

3.4 Implementation details

3.4.1 Gradual Layer Unfreezing for Representation Refinement

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 [46].

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

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	128
Batch size	128
Learning rate λ	2×10^{-5}
Max gradient norm	1.0
Random seed	42

gradient propagation and preserves general knowledge encoded in the backbone model.

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:

$$\mathscr{L}^{(e)}_{\text{active}} = \{\ell_{L-n(e)+1}, ..., \ell_L\} \tag{2}$$

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.

Overall, QoSBERT introduces moderate additional computational overhead compared to standard Transformer-based regression models due to the use of MC Dropout. However, 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.

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:

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.

(1) Semantic Feature Construction. Each QoS record originally consists of basic invocation information and limited numerical features regarding users and services. 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.

(2) Data Sparsity Emulation. 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:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3}$$

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

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{4}$$

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 these 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 [47]: A scalable matrix factorization method that models user-service interactions probabilistically, suitable for large-scale and sparse QoS datasets.

NCF [48]: 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.

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.

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 effectiveness of our architectural design, we compare QoSBERT with a standard supervised fine-tuning (SFT) baseline. This baseline follows the conventional recipe widely adopted in pre-trained language model (PLM) finetuning: it utilizes only the final-layer `[CLS]` representation followed by a multi-layer perceptron (MLP) regression head, without uncertainty modeling or multi-layer pooling. Despite its simplicity, this approach serves as a strong point of comparison to highlight the design gains of QoSBERT.

There are two core distinctions: (1) Representation aggregation: Unlike SFT, which exclusively relies on the last layer's

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
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
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%**

`[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.

5.3 Uncertainty Analysis and Visualization

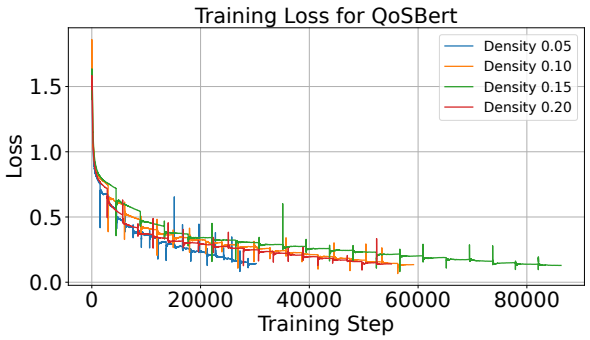
To better understand the predictive uncertainty of our model and its reliability in the QoS prediction task, we conduct a thorough analysis using four complementary visualizations. These visualizations reveal the behavior of uncertainty estimates and validate their calibration quality, supporting the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

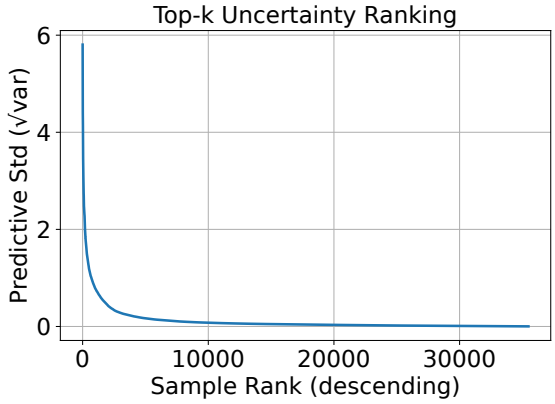
This stratification is critical for practical deployment. In



(a) QoSBERT Training Loss

(b) SFT Training Loss

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



(a) QoSBERT Training Loss

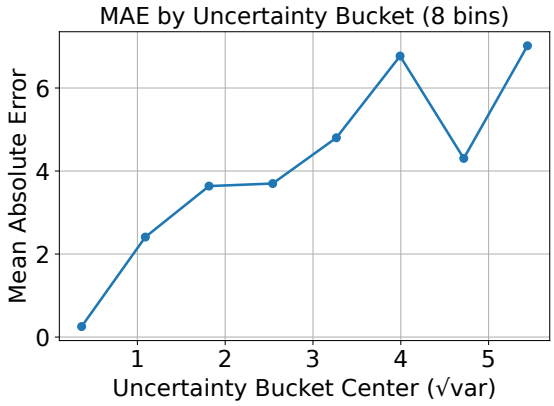
Fig. 3: Training loss curves of QoSBERT and standard SFT under various data densities

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

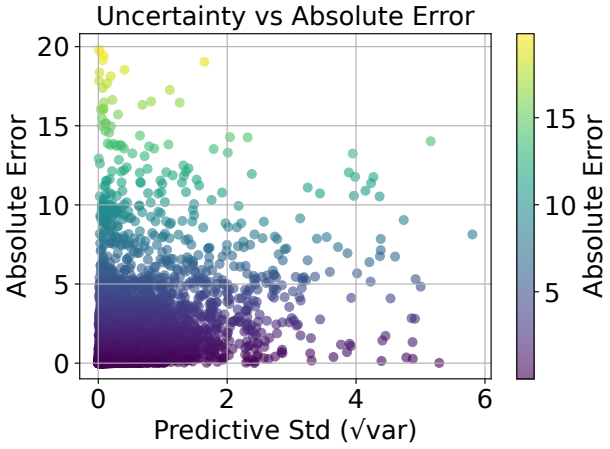
Fig. 4 illustrates the relationship between predicted uncertainty and actual prediction error by grouping samples into 8 equally spaced buckets based on their predicted standard deviation $\sqrt{\text{Var}}$. For each bucket, we compute the average mean absolute error (MAE), revealing how error scales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

A clear positive correlation emerges: the lowest-uncertainty



(a) QoSBERT Training Loss

Fig. 4: Training loss curves of QoSBERT and standard SFT under various data densities



(a) QoSBERT Training Loss

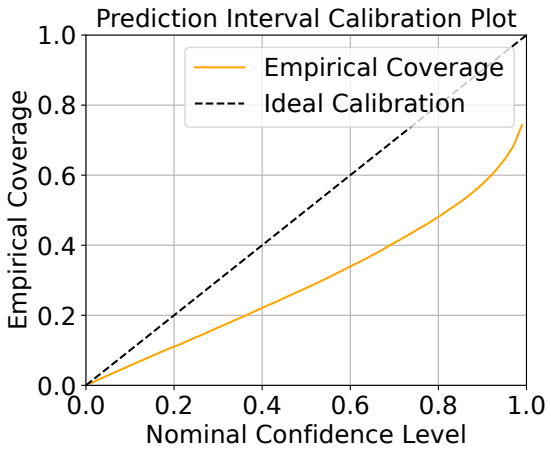
Fig. 5: Training loss curves of QoSBERT and standard SFT under various data densities

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.



(a) QoSBERT Training Loss

Fig. 6: Training loss curves of QoSBERT and standard SFT under various data densities

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.

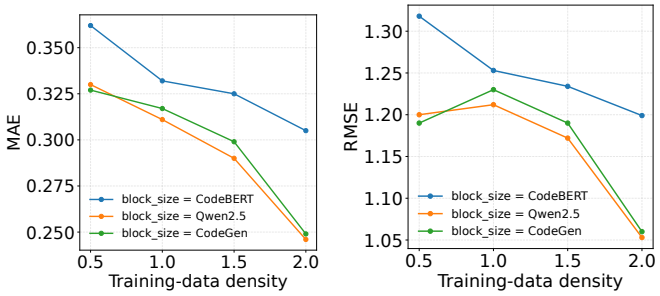
This scatter pattern indicates that while uncertainty is not a perfect proxy for error, it statistically identifies high-risk predictions. For example, by thresholding uncertainty at $\sqrt{\text{Var}} > 0.25$, one can isolate a subset of predictions where the probability of error exceeding 1.0 rises 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., 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.



(a) MAE (b) RMSE

Fig. 7: Effect of the different pretrained

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 [49], Qwen2.5 [22], and CodeGen [44]. The results across multiple data densities are shown in Fig. 8.

These models are selected not only based on their architecture and domain specialization, but also considering practical deployment constraints. In recommendation scenarios, models must meet strict latency and throughput requirements. Models with hundreds of millions of parameters strike a good balance between capacity and inference efficiency, making them ideal candidates for real-time QoS prediction tasks.

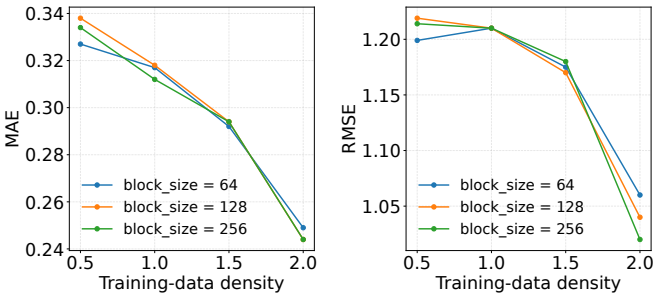
From Fig. 8(a), we observe that Qianwen achieves the lowest MAE across most data densities, especially at high densities where it outperforms CodeBERT by up to 19.3% (D1.4). This suggests that Qianwen may be better adapted to capturing fine-grained user-service interactions due to its advanced pre-training 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. 8(b) further support these findings: both Qianwen and CodeGen consistently outperform CodeBERT, with the largest improvement of 20.1% in RMSE occurring at D1.4 for Qianwen. 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 Qianwen and CodeGen offer a favorable trade-off between prediction accuracy and system efficiency, and thus serve as strong backbones for our QoSBERT.

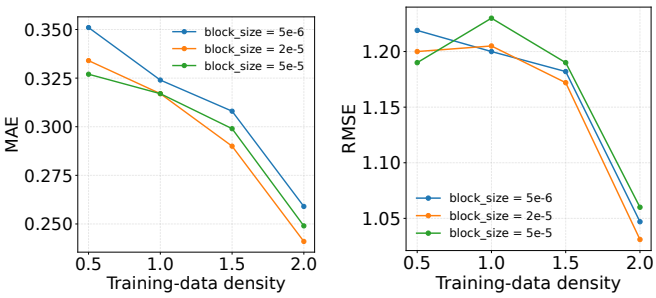
5.5 Ablation experiment

In this section, we conduct ablation experiments to evaluate the effectiveness of our proposed methods. The experiments are divided into two parts: 1) comparing Sparsely Activated Mixture of Experts (MOE) with standard MOE, and 2) analyzing the impact of the parameter ψ on model performance.



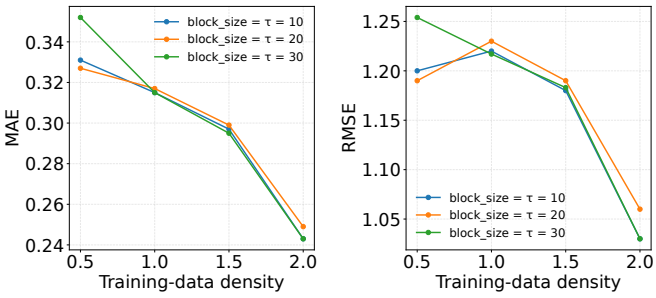
(a) MAE (b) RMSE

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



(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 across different training densities

5.5.1 Impact of block size on Prediction Accuracy

Fig.7 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(0.5), D1.2 (1.0), D1.3 (1.5) and D1.4 (2.0).

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 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. 7 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 $= 128$ 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 $lr = 2 \times 10^{-5}$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, $\tau = 20$ 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.

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.

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] 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.

[44] 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.

[45] 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.

[46] 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.
[47] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," *Advances in neural information processing systems*, vol. 20, 2007.
[48] 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.
[49] 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.