Efficient Nearest Neighbor based Uncertainty Estimation for Natural Language Processing Tasks

Wataru Hashimoto, Hidetaka Kamigaito, Taro Watanabe
Nara Institute of Science and Technology
{hashimoto.wataru.hq3, kamigaito.h, taro}@is.naist.jp

Abstract

Trustworthiness in model predictions is crucial for safety-critical applications in the real world. However, deep neural networks often suffer from the issues of uncertainty estimation, such as miscalibration. In this study, we propose k-Nearest Neighbor Uncertainty Estimation (kNN-UE), which is a new uncertainty estimation method that uses not only the distances from the neighbors, but also the ratio of labels in the neighbors. Experiments on sentiment analysis, natural language inference, and named entity recognition show that our proposed method outperforms the baselines and recent density-based methods in several calibration and uncertainty metrics. Moreover, our analyses indicate that approximate nearest neighbor search techniques reduce the inference overhead without significantly degrading the uncertainty estimation performance when they are appropriately combined.

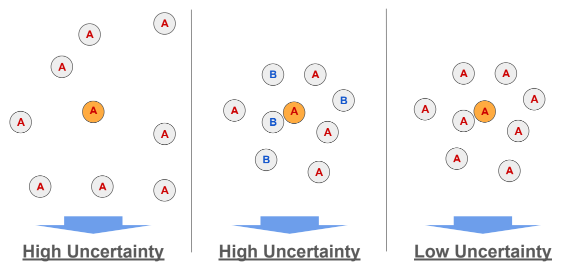

Figure 1: Illustrations of kNN-UE behavior. The orange circle indicates predicted data instances and other circles indicate training data instances. kNN-UE gives high uncertainty when the predicted query representation is far from examples obtained from the kNN search (left) and the predicted label is different from the labels of neighbors (center). kNN-UE outputs low uncertainty only when the query representation is close to neighbors and the labels of neighbors contain many of the model's predicted label (right).

1 Introduction

In order to deploy Deep Neural Networks (DNNs) including Pre-trained Language Models (PLMs) in safety-critical areas, uncertainty estimation (UE) is important. Improving the predictive uncertainty will calibrate the prediction (Guo et al., 2017),[1] or enhance the selective prediction performance which reduces incorrect predictions by providing the option to abstain from the model prediction (Galil et al., 2023). On the other hand, DNNs often fail to quantify the predictive uncertainty, for example, causing miscalibrated prediction (Guo et al., 2017). Such UE performance problems can be mitigated by the PLMs, such as BERT (Devlin et al., 2019) or DeBERTa (He et al., 2021b), that are self-trained on vast amounts of data (Ulmer et al., 2022); nevertheless, there remains considerable room for improvement (Desai and Durrett, 2020).

[1] "Calibration" means the confidence of the prediction aligns with its accuracy.

To address the challenge of UE, multiple stochastic inferences such as MC Dropout (Gal and Ghahramani, 2016) and Deep Ensembles (Lakshminarayanan et al., 2017) are generally effective. On the other hand, these methods require multiple stochastic inferences for a single data instance, which leads to high computational cost, and makes them impractical for real world application. To balance reasonable predictive uncertainty with computational efficiency, Temperature Scaling (Guo et al., 2017) , which scales logits by a temperature parameter, is commonly employed. Furthermore, density-based methods, such as Density Softmax (Bui and Liu, 2024) and Density Aware Calibration (DAC) (Tomani et al., 2023), have demonstrated promising UE performance and inference costs by adjusting model outputs based on estimated density.

However, both Density Softmax and DAC only use the density of the training data. Relying on density alone can sometimes lead to overconfident predictions, even when such confidence is unwar-

Findings of the Association for Computational Linguistics: NAACL 2025, pages 4350–4366

ranted. For instance, the neighbor of the input may contain many examples with labels that differ from the predicted label. In this situation, the prediction should obviously not be trusted. Therefore, we hypothesized that considering both the density and the label information of the neighbors will improve UE performance.

In this study, we propose k-Nearest Neighbor Uncertainty Estimation (kNN-UE), a new density-based UE method that reflects nearest neighbor labels. As illustrated in Figure 1, kNN-UE is designed to achieve the highest prediction confidence when the input and the nearest neighbors are both close in distance and share the same label as the predicted label. Our method weights logits according to the score from the distance between the input example and its neighbors in the datastore created by the training data and the ratio of the model's predicted label matched with the labels in the neighbors. In addition, our method requires only a single forward inference of the model with almost no additional computational cost. The contributions of this research are as follows.

First, our experiments show that kNN-UE improves the UE performance of existing baselines in sentiment analysis, natural language inference, and named entity recognition in both in-domain and out-of-domain settings by combining neighbor label information and distances from neighbors. On the other hand, we also find that naive kNN-UE makes less efficient for token-level tasks such as *sequence-labeling* based named entity recognition due to the execution of kNN to each token.

Second, to mitigate the above latency problem in kNN-UE, we show that approximate kNN search or dimension reduction in kNN-UE improves the inference speed without degrading UE performance much more, while combining them leads to degrading the UE performance.

Our code is available at `https://github.com/wataruhashimoto52/knn_ue`.

2 Related Work

Uncertainty Estimation for Natural Language Processing Tasks Studies about UE for NLP tasks are limited when compared with those for image datasets. Kotelevskii et al. (2022) has shown excellent performance in classification with rejection tasks and out-of-distribution detection tasks using uncertainty scores using density estimation results. Vazhentsev et al. (2022) performed misclassification detection using Determinantal point processes (Kulesza and Taskar, 2012), spectral normalization, Mahalanobis distance and loss regularization in text classification and NER. However, these are still focusing only on the feature representation or the density, not the labels of the neighbors. He et al. (2024) proposed a framework that considers uncertainty between tokens in NER. However, the target task is limited to NER, and it is not for confidence calibration. Hashimoto et al. (2024) shows that simple data augmentation methods in NER can improve UE performance without additional inference costs, but its effectiveness is limited in the in-domain. Our kNN-UE improves the UE performance in the in-domain and the out-of-domain classification and NER tasks using not only kNN density but also neighbor labels.

k-Nearest Neighbor Language Models / Machine Translation k-Nearest Neighbor Language Model (kNN-LM) (Khandelwal et al., 2020) has been proposed, which performs linear interpolation of kNN probability based on distance from neighbors and base model probability, in the language modeling task. k-Nearest Neighbor Machine Translation (kNN-MT) applied the kNN-LM framework to machine translation (Khandelwal et al., 2021). kNN-LM and kNN-MT have been successful because they enhance predictive performance through the memorization and use of rich token representations of pre-trained language models and mitigate problems such as a sparsity comes from low-frequency tokens (Zhu et al., 2023). The main issue on kNN-LM and kNN-MT is the inference overhead, and there are several studies to solve this problem. He et al. (2021a) employs datastore compression, adaptive retrieval, and dimension reduction to reduce computational overhead with retaining perplexity. Deguchi et al. (2023) dramatically improves decoding speed by dynamically narrowing down the search area based on the source sentence. We investigate that whether UE performance in kNN-UE can keep or not with reducing inference time by introducing some of the speed-up techniques established in kNN-LM/MT.

3 Preliminary

3.1 Definitions

In multiclass classification, we assume a dataset $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ consisting of N examples, where $y_n \in \{1, 2, \ldots, J\}$ denotes its corresponding class label among J possible classes.[2] We use the trained neural network feature extractor f and the classifier g for classification, where $f(\boldsymbol{x}) \in \mathbb{R}^D$. g gives us the logits $\boldsymbol{z} = g(f(\boldsymbol{x}))$ and we obtain the confidence $\boldsymbol{p} = \mathrm{softmax}(\boldsymbol{z})$.

3.2 Density Softmax

Density Softmax (Bui and Liu, 2024) obtains confidence by weighting logits with normalized log-likelihood from a trained density estimator. $\boldsymbol{\beta}$ are the parameters of the density estimator; $p(f(\boldsymbol{x}); \boldsymbol{\beta})$ is the normalized log-likelihood from the density estimator, then the corrected confidence is written as

$$p(y_i|\boldsymbol{x}) = \frac{\exp\left(p(f(\boldsymbol{x}); \boldsymbol{\beta}) \cdot z_i\right)}{\sum_{j=1}^{J} \exp\left(p(f(\boldsymbol{x}); \boldsymbol{\beta}) \cdot z_j\right)}. \quad (1)$$

In Density Softmax, the closer the normalized log-likelihood to zero, the closer the prediction to Uniform distribution. Density Softmax achieves reasonable latency and competitive UE performance with state-of-the-art methods at the cost of demanding the density estimator training and multiple base model training.[3]

3.3 Density Aware Calibration (DAC)

DAC is a confidence calibration method using multiple feature representations, which is similar to the kNN-based out-of-distribution detection (Sun et al., 2022). DAC (Tomani et al., 2023) scales the logits by using sample-dependent temperature $\Phi(\boldsymbol{x}, \boldsymbol{w})$

$$p(y_i|\boldsymbol{x}) = \frac{\exp\left(z_i/\Phi(\boldsymbol{x}, \boldsymbol{w})\right)}{\sum_{j=1}^{J} \exp\left(z_j/\Phi(\boldsymbol{x}, \boldsymbol{w})\right)} \quad (2)$$

where

$$\Phi(\boldsymbol{x}, \boldsymbol{w}) = \sum_{l=1}^{L} w_l s_l + w_0. \quad (3)$$

$\boldsymbol{w} \in w_1 ... w_L$ are the weights for every layer of the base model, s_l is the averaged distance from kNN search on l-th layer, and w_0 is the bias term. $w_0 ... w_L$ are optimized using the L-BFGS-B method (Liu and Nocedal, 1989) based on the loss in the validation set. In the original DAC paper, the UE performance tends to improve with the increase in the number of layer representation (Tomani et al., 2023). Therefore, we use all the hidden representations in each layer of the base PLMs.

DAC is a non-parametric method that makes not assumptions about the training data distribution unlike Density Softmax (Bui and Liu, 2024), which relies on some density estimators. On the other hand, the recent kNN-based DAC still relies only on the distances to the neighbors. These methods do not take into account the label information of the input neighbors, which limits the improvement of the UE performance.

4 Proposed Method: k-Nearest Neighbor Uncertainty Estimation (kNN-UE)

The main idea of our proposed method, kNN-UE, stems from the notion that the density-based UE methods can be further improved by using label information about the training data instances that make up the density.

In order to take into account the variance of neighbor labels, our kNN-UE explicitly includes the label agreement information of the predicted instance and its neighbor examples when calculating the confidence. More specifically, we regard the prediction as more reliable only when the prediction is in a region where training data is dense and the predicted label and the labels of the data instances that make up the dense region are mostly the same, as illustrated in the right part of Figure 1. Otherwise, for example, if there is a lot of discrepancy in the neighbor labels and the predicted label, we treat the prediction as unreliable, indicated in the middle of Figure 1.

In our kNN-UE, we introduce two terms: one related to the density of the training data and one related to the degree of agreement of the predicted data and neighbor labels. Confidence of i-th label obtained by kNN-UE is following the formula:

$$p(y_i|\boldsymbol{x}) = \frac{\exp(W_{k\mathrm{NN}}(\hat{y}) \cdot z_i)}{\sum_{j=1}^{J} \exp(W_{k\mathrm{NN}}(\hat{y}) \cdot z_j)} \quad (4)$$

where

$$W_{k\mathrm{NN}}(\hat{y}) = \underbrace{\frac{\alpha}{K} \sum_{k=1}^{K} \exp\left(-\frac{d_k}{\tau}\right)}_{\text{distance term}} + \underbrace{\lambda\left(\frac{S(\hat{y})}{K} + b\right)}_{\text{label term}}. \quad (5)$$

[2]In the case of sequence labeling, we can interpret the number of data N as the product of the raw number of data instances and the sequence length.

[3]Details for the density estimator in this study are in Appendix B.

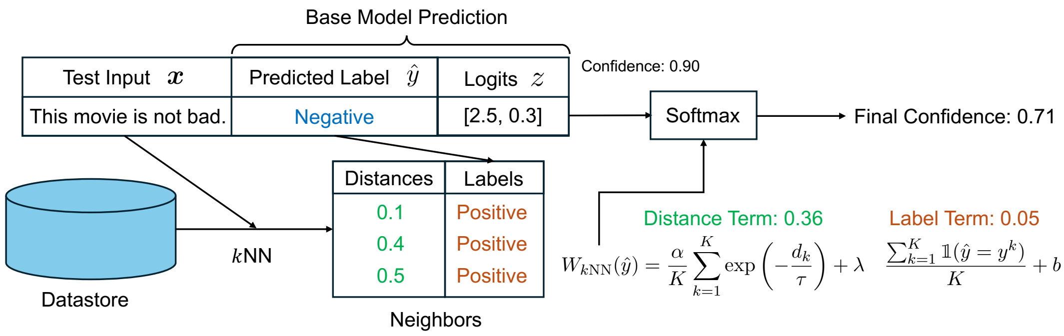

Figure 2: A diagram of kNN-UE when $K = 3$ and the estimated hyperparameters are $\alpha = 0.5$, $\tau = 1.0$, $\lambda = 0.5$ and $b = 0.1$. A datastore is constructed with the representations of the training data as keys and their labels as values. The distances of the nearest examples from the test representation, and the neighbor labels are aggregated into $W_{k\mathrm{NN}}(\hat{y})$. Finally we obtain calibrated confidence by correcting the raw logits with $W_{k\mathrm{NN}}(\hat{y})$ as in Eq. 4.

K is the number of neighbors from kNN search, $S(\hat{y}) = \sum_{k=1}^{K} \mathbb{1}(\hat{y} = y^k)$ is the count when the predicted label $\hat{y}$ and the label of the k-th neighbor y^k is same, d_k is the distance between the k-th $f(\boldsymbol{x})$ representation obtained by kNN search and the representations of training data.[4] The parameters α, τ, $\lambda \in \mathbb{R}_+$ and $b \in \mathbb{R}$ are optimized using the L-BFGS-B method based on the loss in the validation set.

When the distance and label terms are smaller and $W_{k\mathrm{NN}}(\hat{y})$ is closer to zero, the closer the prediction is to Uniform distribution, which allows us to better estimate the confidence of the prediction. In this study, we also conduct experiments without the label term in Equation 5, to emphasize the importance of kNN neighbor labels in UE. We summarize a diagram of kNN-UE in Figure 2.

5 Experimental Settings

5.1 Tasks and Datasets

We measure the UE performance on Sentiment Analysis (SA), Natural Language Inference (NLI), and Named Entity Recognition (NER) in In-domain (ID) and Out-of-Domain (OOD) settings.[5] Dataset statistics are described in Appendix A.

Sentiment Analysis (SA) is a task to classify whether the text sentiment is positive or negative. The IMDb movie review dataset (Maas et al., 2011) is treated as ID, and the Yelp restaurant review dataset (Zhang et al., 2015) is treated as OOD.

Natural Language Inference (NLI) classifies the relationship between a hypothesis sentence and a premise sentence. We treat the Multi-Genre Natural Language Inference (MNLI) dataset (Williams et al., 2018) as ID and the Stanford Natural Language Inference (SNLI) dataset (Bowman et al., 2015) as OOD.

Named Entity Recognition (NER) extracts the named entities, such as a person, organization, or location. The NER task was carried out in the framework of *sequence labeling*. We regard the OntoNotes 5.0 dataset (Pradhan et al., 2013) broadcast news (`bn`) domain as ID, and newswire (`nw`) and telephone conversation (`tc`) domains as OOD.

5.2 Existing Methods

We employ the simple baselines: Softmax Response (SR) (Cordella et al., 1995), Temperature Scaling (TS) (Guo et al., 2017), Label Smoothing (Miller et al., 1996; Pereyra et al., 2017) and MC Dropout (Gal and Ghahramani, 2016). In addition, we use the recent strong baselines for UE: Spectral-Normalized Gaussian Process (SNGP) (Liu et al., 2020), Posterior Networks (PN) (Charpentier et al., 2020), Mahalanobis Distance with Spectral-Normalized Network (MDSN) (Vazhentsev et al., 2022), E-NER (Zhang et al., 2023), Density Softmax (Bui and Liu, 2024), and DAC (Tomani et al., 2023). Details on baselines can be found in Appendix C. We have also experimented with a variant of kNN-UE without the label term in Eq. 5, denoted by "w/o label" to emphasize the impact of the neighbor labels.

[4] Note that kNN-UE is also "accuracy-preserving" same as DAC because $W_{k\mathrm{NN}}(\hat{y})$ is a scalar, not a class-wise score.

[5] The datasets in SA and NLI were set up with reference to Xiao et al. (2022).

5.3 Training Settings

In all experiments, we train and evaluate the models on a single NVIDIA A100 GPU with 40GB of memory. We used $\mathrm{DeBERTaV3_{BASE}}$[6] and $\mathrm{mDeBERTaV3_{BASE}}$[7] (He et al., 2023), as the Transformer encoder from `transformers` (Wolf et al., 2020) pre-trained model checkpoints. We use the cross-entropy loss in all experiments, including the optimization of hyperparameters in kNN-UE. Batch size is 32, and the initial learning rate was set to 1e-5. The gradient clipping is applied with the maximum norm of 1. All experiments are run five times, and we report the mean and standard deviation of the scores.

Datastore Construction It is necessary to maintain the representation of the data for training a density estimator in Density Softmax and kNN search in DAC and kNN-UE. We use the final layer representations corresponding to CLS tokens in SA and NLI. In NER, we stored the hidden representation of the final layer as a token representation corresponding to the beginning of the word.

k-Nearest Neighbor Search We use `faiss` (Douze et al., 2024) as the GPU-accelerated kNN search toolkit. Unless otherwise specified, we fix the number of neighbors $K = 32$ in kNN search,[8] and use `faiss.IndexFlatL2` which is an index for exact search in L2 norm, as the default in kNN-UE.

5.4 Evaluation

To evaluate the confidence calibration performance, we choose *Expected Calibration Error* (ECE) and *Maximum Calibration Error* (MCE) (Naeini et al., 2015). For selective prediction, we evaluate *Area Under the Receiver Operator Characteristic curve* (AUROC) and *Excess-Area Under the Risk-Coverage curve* (E-AURC) (Geifman et al., 2019). Evaluation metrics computation details are described in Appendix D. In NER, we performed the evaluation of the UE performance with the flat recombination of the labels and the confidence for all tokens, respectively.

[6] `https://huggingface.co/microsoft/deberta-v3-base`

[7] `https://huggingface.co/microsoft/mdeberta-v3-base`

[8] In Section 7.1, we conducted experiments to examine the behavior when varying K over the set $\{8, 16, 32, 64, 128\}$, with $K = 32$ representing the median.

6 Results

6.1 Sentiment Analysis

In SA, we evaluate the confidence calibration, selective prediction and out-of-distribution detection performance.

Confidence Calibration and Selective Prediction First, we present the UE results for sentiment analysis by differentiating the in-domain and out-of-main performance in Table 1. kNN-UE consistently outperforms existing methods in terms of ECE, MCE, and E-AURC. In AUROC, LS outperforms in OOD setting, but kNN-UE outperforms existing methods in ID setting. Furthermore, the proposed method clearly outperforms DAC that uses neighbor search results for each hidden representation with the additional label term. The lower UE performance than kNN-UE in DAC is probably due to the difficulty in optimizing hyperparameters by comprising many layers.

Out-of-Distribution Detection Following the previous study (Tomani et al., 2023), we carried out the experiments in the out-of-distribution detection task, which determines whether a data instance is in-domain or not. This task is based on the intuition that we want to return predictions with high confidence in ID but with low confidence in predictions in OOD. We evaluated the out-of-distribution detection performance by using maximum softmax probability as the uncertainty score, and report FPR@95 (the FPR when the TPR is 95%), AUROC, Area Under the Precision-Recall curve (AUPR)-in and AUPR-out. AUPR-in indicates the AUPR score when ID samples are treated as positive; AUPR-out is vice versa.

Table 3 shows the out-of-distribution detection results when using IMDb/Yelp datasets as ID/OOD, respectively, in $\mathrm{mDeBERTaV3_{BASE}}$ model. kNN-UE consistently shows the out-of-distribution detection performance improvement.

6.2 Natural Language Inference

We show the results of in-domain and out-of-domain UE in NLI task using the DeBERTaV3 model in Table 2. Similar to Section 6.1, kNN-UE shows the best UE performance, especially when the label term is included. Galil et al. (2023) have reported that improving calibration performance does not necessarily lead to the improved selective prediction performance, but our proposed method improves both types of metrics. On the other hand,

Methods	IMDb (In-domain)				Yelp (Out-of-domain)			
	ECE (↓)	MCE (↓)	AUROC (↑)	E-AURC (↓)	ECE (↓)	MCE (↓)	AUROC (↑)	E-AURC (↓)
SR	4.42±0.41	24.06±3.52	98.35±0.10	10.60±2.81	4.69±1.20	21.02±6.74	98.15±0.39	11.84±3.15
TS	4.10±0.31	20.43±5.01	98.45±0.21	11.36±2.82	5.10±1.19	19.70±1.35	98.20±0.46	12.91±4.12
LS	1.88±0.41	21.50±4.53	98.36±0.45	14.52±7.24	2.53±0.43	16.47±3.51	**98.30±0.45**	12.90±6.09
MC Dropout	4.28±0.27	23.74±3.52	98.57±0.12	9.17±1.74	4.33±0.54	20.17±2.79	98.28±0.25	10.01±2.01
SNGP	4.18±0.30	22.69±4.83	98.53±0.15	9.95±1.17	4.89±0.59	21.28±4.68	98.10±0.27	11.42±2.14
PN	4.28±0.43	24.43±0.20	98.06±0.27	10.99±5.63	4.69±0.35	24.41±0.32	97.56±0.25	15.82±3.94
MDSN	4.45±0.43	23.97±5.05	98.48±0.08	10.25±0.86	5.32±0.92	21.33±2.91	98.00±0.20	11.12±3.53
Density Softmax	4.23±0.36	27.10±6.92	98.34±0.08	11.39±2.48	4.99±0.48	21.98±3.68	98.09±0.24	13.05±2.72
DAC	1.51±0.33	14.17±2.73	98.36±0.37	12.72±6.15	2.35±0.12	6.44±2.23	97.86±0.60	14.26±5.90
kNN-UE (w/o label)	1.33±0.36	13.13±3.24	**98.65±0.13**	9.36±0.36	2.23±0.29	6.33±2.76	98.27±0.11	10.97±0.91
kNN-UE	**0.95±0.12**†	**9.02±1.39**†	98.64±0.12	**7.97±0.61**†	**1.45±0.15**†	**4.17±1.52**	98.23±0.39	**9.92±0.61**

Table 1: ECE, MCE, AUROC, and E-AURC results about SA task on IMDb (In-domain) and Yelp (Out-of-domain) for $\mathrm{mDeBERTaV3_{BASE}}$ model. Bolds indicate the best result. † indicates significantly improved than existing methods ($p < 0.05$) by using t-test.

Methods	MNLI (In-domain)				SNLI (Out-of-domain)			
	ECE (↓)	MCE (↓)	AUROC (↑)	E-AURC (↓)	ECE (↓)	MCE (↓)	AUROC (↑)	E-AURC (↓)
SR	8.36±0.61	37.61±7.53	97.03±0.12	31.29±2.23	9.77±0.55	36.61±14.05	96.07±0.17	37.62±0.67
TS	2.73±1.86	15.81±11.05	97.06±0.02	31.24±1.86	3.92±1.79	18.13±10.69	96.08±0.13	38.40±2.06
LS	2.89±0.14	28.64±7.90	96.56±0.55	37.98±12.64	3.97±0.45	23.18±6.17	95.61±0.40	44.18±9.18
MC Dropout	8.13±0.65	30.17±6.83	96.97±0.06	32.31±2.25	9.62±0.53	28.90±5.03	96.10±0.11	37.19±2.99
SNGP	10.45±0.56	35.42±13.89	95.91±0.12	42.03±2.72	14.28±1.04	31.16±3.42	93.40±0.44	63.21±6.84
PN	33.83±0.51	37.10±0.71	96.96±0.10	26.33±1.22	32.01±0.61	35.37±0.58	95.57±0.29	40.94±4.49
MDSN	8.34±0.46	29.04±6.43	97.07±0.14	32.03±2.29	9.44±0.47	38.59±13.94	96.11±0.12	38.91±3.06
Density Softmax	8.42±0.43	36.20±5.78	97.03±0.10	32.56±3.29	10.09±0.40	33.59±4.57	95.96±0.19	41.43±2.25
DAC	1.42±0.30	18.79±10.81	96.92±0.10	33.89±2.60	2.27±0.16	11.55±3.48	96.08±0.07	40.23±3.00
kNN-UE (w/o label)	**1.28±0.43**	16.53±11.45	97.09±0.10	30.22±2.80	2.12±0.36	10.00±6.07	**96.12±0.16**	37.33±4.70
kNN-UE	1.41±0.47	**10.77±2.34**†	**97.18±0.09**	**23.83±1.29**†	**1.80±0.37**	**5.12±1.47**†	96.00±0.22	**34.97±2.48**

Table 2: ECE, MCE, AUROC, and E-AURC results about NLI task on MNLI (In-domain) and SNLI (Out-of-domain) for $\mathrm{DeBERTaV3_{BASE}}$ model.

Methods	FPR@95 (↓)	AUROC (↑)	AUPR-In (↑)	AUPR-Out (↑)
SR	82.51±9.49	63.18±5.14	69.51±2.57	54.70±8.48
TS	83.12±7.50	65.63±3.64	70.99±2.02	56.19±6.11
LS	86.88±4.27	62.17±2.83	69.50±1.51	51.38±3.81
MC Dropout	87.33±3.38	63.96±4.09	70.13±2.39	53.18±5.41
SNGP	81.92±3.46	63.27±3.07	68.83±2.10	55.91±3.20
PN	82.84±5.11	67.54±4.29	66.59±2.45	55.32±5.26
Density Softmax	87.54±3.14	58.73±4.33	67.34±2.57	49.19±4.36
DAC	84.98±4.19	64.65±6.18	70.69±3.59	54.81±7.29
kNN-UE (w/o label)	75.87±2.16	70.44±1.70	**74.77±1.44**†	63.39±2.24
kNN-UE	**73.55±5.01**†	**71.11±2.92**†	73.80±2.19	**65.01±3.45**†

Table 3: Out-of-distribution detection results on $\mathrm{mDeBERTaV3_{BASE}}$ model using IMDb/Yelp Polarity as ID/OOD datasets, respectively.

the degree of improvement is larger for calibration performance. Specifically, the largest improvement is obtained on SNLI, where kNN-UE reduces MCE by more than 31.49 % compared to SR. Additional experimental results on the Brier score can be found in Appendix E.

6.3 Named Entity Recognition

To evaluate NLP tasks other than simple multi-class classification, we evaluate kNN-UE in NER. Since NER focuses on entities, we use the product of the confidence of the tokens that construct a single entity as the confidence of the entity.

Table 4 shows the results of in-domain and out-of-domain UE using the OntoNote 5.0 dataset in $\mathrm{mDeBERTaV3_{BASE}}$. kNN-UE shows the best performance in 4 cases, i.e., ECE or MCE, often resulting in large improvements over SR. On the other hand, E-AURC in NER is consistently better without using the kNN-UE label term.[9] E-NER, a recent UE method specifically designed for NER, is close to kNN-UE in its entity level selective prediction performance, but the calibration performance is not high.

kNN-UE shows good UE performance even when the target domain is relatively far from source domain bn, such as tc. We have hypothesized that kNN-UE might not work if the prediction target is too far from the training data distribution. If the prediction target is too far from the training data, the representation of the prediction from the model will be unreliable when compared to the prediction in the same domain as the training data. In general, methods based on feature distances assume that they maintain information relevant to the correctness of the prediction (Postels et al., 2022). Our experiments have shown that the problem could be

[9] Label imbalance or large number of class can significantly affect E-AURC on NER when using kNN-UE with the label term. Details are in Appendix F.

Methods	bn (In-domain)			nw (Out-of-domain)			tc (Out-of-domain)		
	ECE (↓)	MCE (↓)	E-AURC (↓)	ECE (↓)	MCE (↓)	E-AURC (↓)	ECE (↓)	MCE (↓)	E-AURC (↓)
SR	7.79±0.53	50.07±24.15	21.90±1.31	17.05±0.69	37.06±3.13	81.49±4.17	21.20±2.03	42.60±5.84	76.05±5.72
TS	5.34±0.43	75.71±21.96	19.63±1.22	12.76±0.62	26.57±3.97	72.90±4.72	19.69±0.95	47.72±7.34	71.87±8.83
LS	6.46±0.74	50.99±26.73	24.93±1.19	14.78±0.61	30.54±2.84	81.50±6.98	20.99±2.16	65.40±17.16	76.65±7.33
MC Dropout	6.76±0.64	53.13±26.07	19.91±3.39	15.27±1.01	33.60±4.93	77.21±3.72	21.93±1.63	56.56±12.32	75.68±9.30
E-NER	7.98±0.42	61.87±27.06	19.44±1.81	17.42±0.88	40.46±5.33	74.32±4.47	25.42±2.09	59.16±10.33	72.00±6.57
Density Softmax	7.32±0.25	59.05±27.76	25.17±2.63	16.10±0.62	44.66±21.67	80.14±8.50	24.40±1.84	62.50±10.46	80.06±6.27
DAC	**1.62±0.42**	42.96±28.25	21.47±2.90	7.91±0.75	25.28±5.15	75.24±2.43	14.42±1.57	47.92±20.98	80.72±8.19
kNN-UE (w/o label)	3.37±0.71	33.15±3.65	**17.63±0.66**†	8.78±0.62	24.91±1.81	**70.10±4.03**	14.61±0.67	**35.26±7.16**†	**65.41±8.11**
kNN-UE	1.78±0.32	**26.02±13.72**	20.14±1.27	**7.50±0.42**	**16.53±2.61**†	74.27±5.43	**14.15±0.33**	39.84±6.02	71.81±9.04

Table 4: ECE, MCE, and E-AURC results about NER on OntoNotes 5.0 dataset for $\mathrm{mDeBERTaV3_{BASE}}$ model.

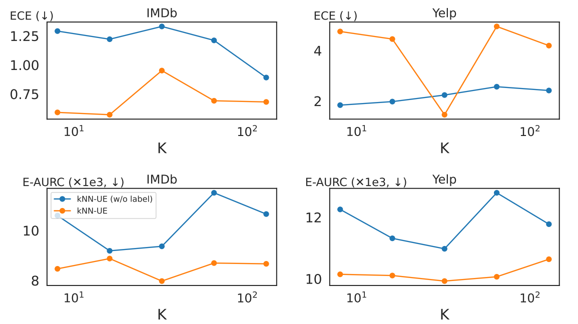

Figure 3: Changes in ECE and E-AURC in SA when changing the number of neighbors of kNN-UE. On the x-axis, the parameter $K \in \{8, 16, 32, 64, 128\}$ is represented on a log scale.

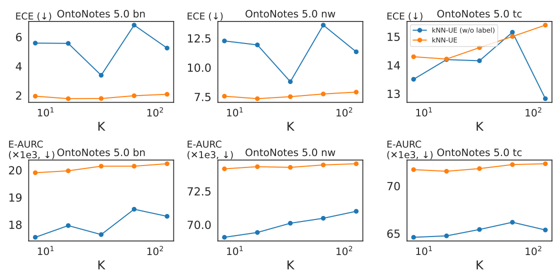

Figure 4: Changes in ECE and E-AURC in NER when changing the number of neighbors of kNN-UE. On the x-axis, the parameters $K \in \{8, 16, 32, 64, 128\}$ are represented on a log scale.

mitigated probably because the domains that the base models do not recognize are limited in the NLP community where there are many strong pre-trained models based on self-supervised learning such as DeBERTaV3.

7 Analysis

7.1 Impact of Top-K

To understand the behavior of kNN-UE, we evaluated the performance in UE when changing the number of neighbors $K \in \{8, 16, 32, 64, 128\}$ during kNN execution.

Scores	Correct Instances		Incorrect Instances	
	kNN-UE (w/o label)	kNN-UE	kNN-UE (w/o label)	kNN-UE
$W_{k\mathrm{NN}}$	0.49	0.50	0.41	0.27
Confidence	0.95	0.93	0.82	0.72

Table 5: Averaged $W_{k\mathrm{NN}}$ and confidence scores with and without label term in kNN-UE for correct and incorrect predicted instances when using IMDb as train/validation and Yelp as test, respectively.

Methods	MNLI	OntoNotes 5.0 bn
SR	8.41±0.03	2.49±0.08
TS	8.42±0.07	2.51±0.08
LS	8.44±0.06	2.53±0.03
MC Dropout	157.52±0.51	39.81±0.39
SNGP	10.58±2.09	-
PN	9.11±0.07	-
MDSN	9.65±1.36	-
E-NER	-	2.51±0.12
Density Softmax	8.57±0.06	2.59±0.05
DAC	785.15±6.72	183.46±0.76
kNN-UE (w/o label)	9.05±0.07	4.94±0.10
kNN-UE	9.08±0.10	4.99±0.07

Table 6: Inference time [s] on MNLI test set and OntoNotes 5.0 bn test set.

Figure 3 and 4 show the results for SA and NER, respectively. As is noticeable in NER, the smaller K, the better UE tends to be. These results suggest that our method requires that nearer examples to calibrate confidence, but more distant examples are not important. When calculating $W_{kNN}(\hat{y})$ in Eq. 5, automatically adjusting the importance weights based on the order or distance of the retrieved nearest neighbors could further improve UE performance. Similar experimental and theoretical analysis of out-of-distribution detection using only kNN distance also suggests that using k-th example is preferable (Sun et al., 2022). Providing a similar theoretical justification for our kNN-UE is an interesting future direction.

7.2 Importance of Label Term in $W_{k\mathrm{NN}}$

We analyze the impact of the label term Eq. 5 on the kNN-UE confidence computation. We have shown that the UE performance is improved in several experiments. However, it is not obvious

Methods	OntoNotes 5.0 bn (In-domain)				OntoNotes 5.0 nw (Out-of-domain)			
	ECE (↓)	MCE (↓)	E-AURC (↓)	time [s]	ECE (↓)	MCE (↓)	E-AURC (↓)	time [s]
SR	7.79±0.53	50.07±24.15	21.90±1.31	2.49±0.08	17.05±0.69	37.06±3.13	81.49±4.17	5.75±0.27
kNN-UE (w/o label)	3.37±0.71	33.15±3.65	17.63±0.66	4.94±0.10	8.78±0.62	24.91±1.81	70.10±4.03	10.36±0.21
kNN-UE	1.78±0.32	26.02±13.72	20.14±1.27	4.99±0.07	7.50±0.42	16.53±2.61	74.27±5.43	10.48±0.12
+ PQ	1.96±0.31	31.33±18.74	20.23±1.27	3.32±0.05	7.57±0.45	16.43±2.73	74.38±5.36	7.23±0.16
+ IVF	1.92±0.31	28.55±11.24	20.13±1.22	3.31±0.06	7.60±0.41	17.12±2.35	74.34±5.35	7.33±0.21
+ DR	2.14±0.37	33.52±10.84	20.12±1.26	2.87±0.04	8.08±0.53	24.03±5.46	74.50±5.42	6.20±0.20

Table 7: ECE, MCE, E-AURC and inference time results about NER on OntoNotes 5.0 bn (In-domain) and OntoNotes 5.0 nw (Out-of-domain) for $\mathrm{mDeBERTaV3_{BASE}}$ model when applied PQ, IVF, and dimension reduction sequentially. DR indicates dimension reduction. For comparison, we also present the results when dimension reduction is only applied to kNN-UE.

Methods	OntoNotes 5.0 bn	OntoNotes 5.0 nw
kNN-UE	100.0	100.0
+ PQ	21.30	51.68
+ IVF	18.60	11.04
+ DR	0.02	0.04
Only DR	43.98	20.35

Table 8: Coverages when PQ, clustering, and PCA are applied sequentially to the example indices obtained by default kNN-UE. Results when applying dimension reduction by PCA individually are also presented for reference.

whether the improvement in UE performance is due to the reduction in $W_{k\mathrm{NN}}$ primarily caused by adding the label term. Therefore, we examined the $W_{k\mathrm{NN}}$ values for correctly and incorrectly predicted instances in both the absence and presence of the label term in kNN-UE. Table 5 shows the distance terms, label terms and $W_{k\mathrm{NN}}$ results for each case. If the predictions are correct, the growth of $W_{k\mathrm{NN}}$ due to the label term is limited. On the other hand, kNN-UE with label term remarkably reduce $W_{k\mathrm{NN}}$ leading to the reduced confidence when the predictions are incorrect. This result suggests that the improvement of the evaluation metrics in kNN-UE with label term is not achieved by increasing the confidence when the prediction is correct, but by appropriately reducing the confidence when the prediction is incorrect.

7.3 Impact of Efficient Nearest Neighbor Search Techniques

We investigate the inference time and UE performance when applying approximate nearest neighbor search techniques and dimension reduction when executing kNN search in kNN-UE as a real world application. As shown in Table 6,[10] in the *sequence labeling* based NER, which requires executing kNN searches per token, it takes twice as much inference time as SR.[11] On the other hand, in k-Nearest Neighbor Language Model (kNN-LM) (Khandelwal et al., 2020), dimension reduction and approximate kNN search techniques are effective to improve inference speed while maintaining perplexity in text generation (He et al., 2021a; Xu et al., 2023). Therefore, inspired by these works for faster kNN-LM, we investigate how the approximate nearest neighbor search techniques, such as Product Quantization (Jégou et al., 2011) (PQ), Inverted File (IVF) clustering and dimension reduction affect the UE and inference speed of our kNN-UE. Description of approximate nearest neighbor search techniques and detailed discussion when each method is individually applied to kNN-UE are in Appendix I.

Results of Combination of PQ, IVF and Dimension Reduction We evaluate the UE performance and inference speed when applying PQ, IVF and dimension reduction are applied. Table 7 shows the results on OntoNotes 5.0 bn and nw test sets as ID/OOD, respectively. The detailed discussion when changing the parameters of PQ, clustering and dimension reduction are shown in Appendix I. We can see that ECE and MCE are degraded when PQ, IVF and dimension reduction by PCA are applied simultaneously to kNN-UE.[12] On the other hand, our results show that applying them appropriately such as combining PQ with IVF improve inference time with mitigating the degradation in UE performance (The results with the parameters for PQ or IVF can be found in Appendix I.1 or I.2). To deepen our understanding of the above changes in the behavior of the uncertainty performance due to appling of approximate kNN search techniques

[10] Other results can be found in Appendix G.

[11] Inference times do not increase as dramatically as k-Nearest Neighbor Language Model (Khandelwal et al., 2020) because kNN can be executed in parallel for both classification and NER.

[12] Distance recomputation does not mitigate this behavior, see Appendix J.

or dimension reduction in kNN-UE, we calculated the coverage that how much the indices obtained when using the default exhaustive search are covered when applying PQ, clustering, and dimension reduction sequentially.

Table 8 shows the coverages on OntoNotes 5.0 `bn` and `nw` as ID/OOD settings, respectively. We can see that applying PQ, clustering, and PCA simultaneously hardly covers any of the indices from the default kNN-UE. It is assumed that applying PQ and PCA in the same time leads to coarse distance computation in a single subvector, which would correspondingly degrade the UE performance in kNN-UE. Actually, the experimental results in Table 17 in Appendix I.3 suggest that excessive dimension reduction in distance computation could have a negative impact on the UE performance. On the other hand, if combined with PQ and IVF, or applied PCA individually, some of the ground-truth nearest neighbor examples still exist.

8 Conclusion

In this paper, we proposed kNN-UE, which estimates uncertainty by using the distance to neighbors and labels of neighbors. The experimental results showed that our method showed higher UE performance than existing UE methods in SA, NLI and NER. Furthermore, our analysis of correctly and incorrectly predicted instances suggests that the improvement in kNN-UE is largely due to the reduction in confidence on incorrect instances. In addition, we investigated the effects of efficient neighbor search techniques in kNN-UE to address the degradation of the inference speed in token-level tasks such as NER. As a result, we found that product quantization, clustering, or dimension reduction improves inference speed without degrading the UE much more, unless combining all of them simultaneously.

9 Limitations

In this study, we focused only on the classification-based tasks. On the other hand, taking advantage of the recent growth of Large Language Models, UE in text generation is also attracting attention (Yoshikawa and Okazaki, 2023; Fadeeva et al., 2023; Lin et al., 2024). Therefore, to investigate the effectiveness of kNN-UE in text generation tasks is an interesting direction for future research. Not only that, our proposed method is applicable to more tasks such as image classification.

Furthermore, although kNN-UE only used the representation of the last layer of the base model, exploring for an appropriate representation for UE is a future challenge. Also, to investigate the relationship between the representation quality and in- and out-of-domain UE performance when using smaller pretrained encoders than DeBERTa, such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) is an interesting direction.

Finally, we used ECE and MCE to measure calibration performance. On the other hand, it may be more appropriate to use other metrics to measure calibration performance when the dataset with multiple annotations including human disagreement is available (Baan et al., 2022), where it may be similar to the label disagreement in similar output representations. In Section 7.2, we showed that our kNN-UE with the label term improves in the direction we expected: it reduces confidence much more when the predictions are inaccurate. However, measuring calibration performance on a variety of data with multiple annotations may provide a more interesting insight into the behavior of our proposed method.

Ethical Considerations

In this study, we used existing datasets that have cleared ethical issues following policies of published conferences. Therefore, they do not introduce any ethical problems. On the other hand, we have an ethical consideration about UE. Specifically, decision support systems with machine learning algorithms do not necessarily have a positive effect on performance. Jacobs et al. (2021) showed that collaboration with machine learning models does not significantly improve clinician's treatment selection performance, and that performance is significantly degraded due to the presentation of incorrect recommendations. This problem is expected to remain even if UE methods are applied to machine learning models. In addition, introducing UE methods could conversely lead humans to give overconfidence in machine learning models, resulting in performance degradation.

Acknowledgements

The authors also acknowledge the Nara Institute of Science and Technology's HPC resources made available for conducting the research reported in this paper.

References

Joris Baan, Wilker Aziz, Barbara Plank, and Raquel Fernandez. 2022. Stop measuring calibration when humans disagree. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1892–1915, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Ha Manh Bui and Anqi Liu. 2024. Density-softmax: Efficient test-time model for uncertainty estimation and robustness under distribution shifts. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 4822–4853. PMLR.

Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. 2020. Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts. In *Advances in Neural Information Processing Systems*, volume 33, pages 1356–1367. Curran Associates, Inc.

L.P. Cordella, C. De Stefano, F. Tortorella, and M. Vento. 1995. A method for improving classification reliability of multilayer perceptrons. *IEEE Transactions on Neural Networks*, 6(5):1140–1147.

Hiroyuki Deguchi, Taro Watanabe, Yusuke Matsui, Masao Utiyama, Hideki Tanaka, and Eiichiro Sumita. 2023. Subset retrieval nearest neighbor machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 174–189, Toronto, Canada. Association for Computational Linguistics.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2017. Density estimation using real NVP. In *International Conference on Learning Representations*.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *arXiv preprint arXiv:2401.08281*.

Ekaterina Fadeeva, Roman Vashurin, Akim Tsvigun, Artem Vazhentsev, Sergey Petrakov, Kirill Fedyanin, Daniil Vasilev, Elizaveta Goncharova, Alexander Panchenko, Maxim Panov, Timothy Baldwin, and Artem Shelmanov. 2023. LM-polygraph: Uncertainty estimation for language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 446–461, Singapore. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA. PMLR.

Ido Galil, Mohammed Dabbah, and Ran El-Yaniv. 2023. What can we learn from the selective prediction and uncertainty estimation performance of 523 imagenet classifiers? In *The Eleventh International Conference on Learning Representations*.

Yonatan Geifman, Guy Uziel, and Ran El-Yaniv. 2019. Bias-reduced uncertainty estimation for deep neural classifiers. In *International Conference on Learning Representations*.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR.

Wataru Hashimoto, Hidetaka Kamigaito, and Taro Watanabe. 2024. Are data augmentation methods in named entity recognition applicable for uncertainty estimation? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18852–18867, Miami, Florida, USA. Association for Computational Linguistics.

Jianfeng He, Linlin Yu, Shuo Lei, Chang-Tien Lu, and Feng Chen. 2024. Uncertainty estimation on sequential labeling via uncertainty transmission. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2823–2835, Mexico City, Mexico. Association for Computational Linguistics.

Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. 2021a. Efficient nearest neighbor language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5703–5714, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021b. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Maia Jacobs, Melanie F. Pradier, Thomas H. McCoy, Roy H. Perlis, Finale Doshi-Velez, and Krzysztof Z. Gajos. 2021. How machine-learning recommendations influence clinician treatment selections: the example of the antidepressant selection. *Translational psychiatry*, 11(1).

Herve Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *International Conference on Learning Representations*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *International Conference on Learning Representations*.

Nikita Yurevich Kotelevskii, Aleksandr Artemenkov, Kirill Fedyanin, Fedor Noskov, Alexander Fishkov, Artem Shelmanov, Artem Vazhentsev, Aleksandr Petiushko, and Maxim Panov. 2022. Nonparametric uncertainty quantification for single deterministic neural network. In *Advances in Neural Information Processing Systems*.

Alex Kulesza and Ben Taskar. 2012. *Determinantal Point Processes for Machine Learning*. Now Publishers Inc., Hanover, MA, USA.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, page 6405–6416.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2024. Generating with confidence: Uncertainty quantification for black-box large language models. *Transactions on Machine Learning Research*.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528.

Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. 2020. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. In *Advances in Neural Information Processing Systems*, volume 33, pages 7498–7512. Curran Associates, Inc.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

David J. Miller, Ajit V. Rao, Kenneth M. Rose, and Allen Gersho. 1996. A global optimization technique for statistical classifier design. *IEEE Trans. Signal Process.*, 44:3108–3122.

Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 2901–2907. AAAI Press.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *Proceedings of the Inter- national Conference on Learning Representations (Workshop)*.

Janis Postels, Mattia Segù, Tao Sun, Luca Daniel Sieber, Luc Van Gool, Fisher Yu, and Federico Tombari. 2022. On the practicality of deterministic epistemic uncertainty. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17870–17909. PMLR.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Danilo Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538, Lille, France. PMLR.

Murat Sensoy, Lance Kaplan, and Melih Kandemir. 2018. Evidential deep learning to quantify classification uncertainty. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. 2022. Out-of-distribution detection with deep nearest neighbors. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 20827–20840. PMLR.

Christian Tomani, Futa Kai Waseda, Yuesong Shen, and Daniel Cremers. 2023. Beyond in-domain scenarios: Robust density-aware calibration. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 34344–34368. PMLR.

Dennis Ulmer, Jes Frellsen, and Christian Hardmeier. 2022. Exploring predictive uncertainty and calibration in NLP: A study on the impact of method & data scarcity. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2707–2735, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Artem Vazhentsev, Gleb Kuzmin, Artem Shelmanov, Akim Tsvigun, Evgenii Tsymbalov, Kirill Fedyanin, Maxim Panov, Alexander Panchenko, Gleb Gusev, Mikhail Burtsev, Manvel Avetisian, and Leonid Zhukov. 2022. Uncertainty estimation of transformer predictions for misclassification detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8237–8252, Dublin, Ireland. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yuxin Xiao, Paul Pu Liang, Umang Bhatt, Willie Neiswanger, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2022. Uncertainty quantification with pre-trained language models: A large-scale empirical analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7273–7284, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Frank F. Xu, Uri Alon, and Graham Neubig. 2023. Why do nearest neighbor language models work? In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38325–38341. PMLR.

Hiyori Yoshikawa and Naoaki Okazaki. 2023. Selective-LAMA: Selective prediction for confidence-aware evaluation of language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2017–2028, Dubrovnik, Croatia. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Zhen Zhang, Mengting Hu, Shiwan Zhao, Minlie Huang, Haotian Wang, Lemao Liu, Zhirui Zhang, Zhe Liu, and Bingzhe Wu. 2023. E-NER: Evidential deep learning for trustworthy named entity recognition. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1619–1634, Toronto, Canada. Association for Computational Linguistics.

Wenhao Zhu, Jingjing Xu, Shujian Huang, Lingpeng Kong, and Jiajun Chen. 2023. INK: Injecting kNN knowledge in nearest neighbor machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15948–15959, Toronto, Canada. Association for Computational Linguistics.

A Dataset Statistics

The dataset statistics in our study is shown in Table 9.

Tasks	Datasets	N_{class}	Train	Val	Test
SA	**IMDb**	2	25,000	12,500	12,500
	Yelp	2	-	-	19,000
NLI	**MNLI**	3	392,702	4,907	4,908
	SNLI	3	-	-	9,824
NER	**OntoNotes 5.0** (`bn`)	37	10,683	1,295	1,357
	OntoNotes 5.0 (`nw`)	37	-	-	2,327
	OntoNotes 5.0 (`tc`)	37	-	-	1,366

Table 9: Dataset Statistics. Bolds indicate In-domain.

B Training Settings for Density Estimator in Density Softmax

In Density Softmax (Bui and Liu, 2024), we use RealNVP (Dinh et al., 2017) as the density estimator, which has two coupling structures. Table 10 shows the hyperparameters for training RealNVP as the density estimator in Density Softmax.

Hyperparameters	Values
learning rate	1e-4
optimizer	AdamW (Loshchilov and Hutter, 2019)
early stopping patient	5
number of coupling layers	4
hidden units	16

Table 10: Hyperparameters for RealNVP in Density Softmax.

C Details of Baselines

Softmax Response (SR) is a trivial baseline, which treats the maximum score from output of the base model's softmax layer as the confidence (Cordella et al., 1995).

Temperature Scaling (TS) is a calibration technique by which the logits are divided by a temperature parameter T before applying the softmax function (Guo et al., 2017). We optimized T by L-BFGS on validation set loss.

Label Smoothing (LS) is the calibration and generalization technique by introducing a small degree of uncertainty ϵ in the target labels during training (Miller et al., 1996; Pereyra et al., 2017). In LS, we optimized $\epsilon \in \{0.01, 0.05, 0.1, 0.2, 0.3\}$ by using validation set accuracy when SA and NLI, and validation set F_1 when NER.

MC Dropout is an UE technique by M times stochastic inferences with activating dropout (Gal and Ghahramani, 2016). In our experiments, we set $M = 20$ for all evaluations, and the dropout rate is 0.1.

Spectral-Normalized Gaussian Process (SNGP) uses spectral normalization of the weights for distance-preserving representation and Gaussian Processes in the output layer for estimating uncertainty (Liu et al., 2020).

Posterior Networks (PN) is one of the methods in the Evidential Deep Learning (EDL) framework (Sensoy et al., 2018) that assumes a probability distribution for class probabilities (Charpentier et al., 2020), which uses normalizing flow (Rezende and Mohamed, 2015) to estimate the density of each class in the latent space.

Mahalanobis Distance with Spectral-Normalized Network (MDSN) is a Mahalanobis distance based UE method that benefits from by spectral normalization of the weights (Vazhentsev et al., 2022), similar to SNGP.

E-NER applies EDL framework for NER by introducing uncertainty-guided loss terms (Zhang et al., 2023).

D Details of Evaluation Metrics

Expected Calibration Error (ECE) ECE (Naeini et al., 2015) quantifies the difference between the accuracy and confidence of a model. Formally, ECE is expressed as:

$$\mathrm{ECE} = \sum_{b=1}^{B} \frac{|\mathcal{D}_b|}{n} \left|\mathrm{acc}(\mathcal{D}_b) - \mathrm{conf}(\mathcal{D}_b)\right| \quad (6)$$

where B is the number of confidence interval bins, $\mathcal{D}_b$ denotes the set of examples with predicted confidence scores in the b-th bin, n is the total number of examples, $\mathrm{acc}(\mathcal{D}_b)$ is the accuracy of the model on the examples in $\mathcal{D}_b$, and $\mathrm{conf}(\mathcal{D}_b)$ is the average confidence of the model on the examples in $\mathcal{D}_b$. In this study, we use $B = 10$.

Maximum Calibration Error (MCE) MCE, as detailed by Naeini et al. (2015) measures the maximum difference between the model's accuracy and the confidence across variousb confidence levels. MCE is defined as:

$$\mathrm{MCE} = \max_{b=1}^{B} \left|\mathrm{acc}(\mathcal{D}_b) - \mathrm{conf}(\mathcal{D}_b)\right|, \quad (7)$$

A lower MCE means that there is a small risk that the confidence of the model's prediction will deviate greatly from the actual correct answer. In this study, we use $B = 10$, same as ECE.

Area Under the Risk-Coverage curve (AURC) The AURC is the area of the risk-coverage curve when the confidence levels of the forecasts corresponding to the N data points are sorted in descending order. The larger the area, the lower the error rate corresponding to a higher confidence level, which means that the output confidence level is more appropriate. Formally, AURC is defined as:

$$\mathrm{AURC} = \sum_{n=1}^{N} \frac{\sum_{j=1}^{n} g(x_j)}{i \times N} \quad (8)$$

where $g(x)$ returns 1 if the prediction is wrong and 0 otherwise.

Excess-Area Under the Risk-Coverage curve (E-AURC) E-AURC (Geifman et al., 2019) is a measure of the AURC score normalized by the smallest risk-coverage curve area $\mathrm{AURC}^{\star} \approx \hat{r} + (1 -$

Methods	SA		NLI	
	IMDb	Yelp	MNLI	SNLI
SR	5.00±0.27	5.83±0.98	9.50±0.40	11.02±0.41
TS	5.09±0.42	6.67±1.36	8.31±0.25	9.60±0.21
LS	4.64±0.23	5.16±0.92	8.73±0.23	10.18±0.17
MC Dropout	4.88±0.21	5.45±0.55	9.33±0.36	11.00±0.28
SNGP	4.78±0.15	5.99±0.39	12.25±5.38	13.45±4.57
PN	10.31±0.28	11.16±0.22	20.76±0.32	21.11±0.42
Density Softmax	4.82±0.18	6.05±0.38	9.60±0.34	11.28±0.41
DAC	4.44±0.33	5.44±0.71	8.21±0.25	9.55±0.35
kNN-UE (w/o label)	4.37±0.16	5.10±0.12	8.15±0.15	9.52±0.32
kNN-UE	**4.21±0.14**	**5.02±0.42**	**8.07±0.18**	**9.44±0.28**

Table 11: Brier score results using IMDb/Yelp and MNLI/SNLI as ID/OOD datasets, respectively.

$\hat{r})\ln(1-\hat{r})$, where $\hat{r}$ is the error rate of the model. The reason for normalizing the AURC is that the AURC depends on the predictive performance of the model and allows for performance comparisons of confidence across different models and training methods. E-AURC is defined as:

$$\text{E-AURC} = \text{AURC} - \text{AURC}^{\star} \tag{9}$$

E-AURC scores are reported with multiplying by 1,000 due to visibility.

E Additional Results on the Brier score

The Brier score is a widely used metric in UE community for evaluating the probabilistic predictions. The metric measures the mean squared difference between the predicted probability assigned to the predicted label and the actual outcome. This evaluation serves as a holistic assessment of model performance, reflecting both fit and calibration, in the following formula:

$$\text{Brier score} = \frac{1}{N}\sum_{n=1}^{N}(p_n - o_n), \tag{10}$$

where p_n is the predicted probability assigned to the prediction, and o_n is the actual outcome. Table 11 shows the results on the Brier score. These results indicate kNN-UE improves calibration performance more prominently than other methods while maintaining prediction performance.

F The impact of kNN-UE with label term in NER on E-AURC

NER tasks are often in label imbalanced settings, where the "O" label is typically much more than other entity-related labels. Additionally, in the OntoNotes 5.0 dataset, the number of labels is 37, as shown in Table 9, which is significantly higher than in SA and NLI tasks. As a result, compared to SA and NLI, neighbor labels will contain much more different labels from the predicted label. The presence of many other labels in the neighbors that are different from the predicted label can lead to excessively low confidence in kNN-UE using the label term, even though the prediction is correct because $S(\hat{y})$ in the label term becomes lower in NER. The impact of that bias for calibration errors, such as ECE and MCE, will be limited. However, low confidence in accurate prediction reduces the coverage much in E-AURC, leading to a degradation in E-AURC.

For example, assume that in a test data set of 6 examples for 3 classes classification, the predictions for the first 3 examples are incorrect and the latter 3 examples are correct. In case A, we assume that prediction confidences are [[0.25, 0.25, 0.5], [0.25, 0.25, 0.5], [0.25, 0.25, 0.5], **[0.25, 0.25, 0.5]**, [0.02, 0.02, 0.96], [0.01, 0.01, 0.98]]. In case B, we assume that prediction confidences are [[0.25, 0.25, 0.5], [0.25, 0.25, 0.5], [0.25, 0.25, 0.5], **[0.275, 0.275, 0.45]**, [0.02, 0.02, 0.96], [0.01, 0.01, 0.98]]. In these settings, the ECE and E-AURC for each case are shown in Table 12. These scores indicate that E-AURC is strongly penalized when the confidence in a correct prediction is lower than the confidence in an incorrect prediction.

Case	ECE (↓)	E-AURC (↓)
A	17.67	18.80
B	16.83	121.57

Table 12: ECE and E-AURC in two toy cases of Appendix F.

G Inference Time Full Results

We show the inference time full results on out-of-domain test sets in Table 13.

Methods	SNLI	OntoNotes 5.0 `nw`
SR	21.59±0.76	5.75±0.27
TS	21.64±0.07	5.79±0.17
LS	21.70±0.07	5.80±0.19
MC Dropout	396.86±1.10	101.98±0.83
SNGP	24.59±0.08	-
PN	23.26±0.05	-
MDSN	23.39±0.85	-
E-NER	-	5.78±0.61
Density Softmax	22.02±0.05	6.02±0.07
DAC	2346.62±36.06	326.00±1.41
kNN-UE (w/o label)	23.02±0.04	10.36±0.21
kNN-UE	23.07±0.05	10.48±0.12

Table 13: Inference time [s] on SNLI test set and OntoNotes 5.0 `nw` test set.

Methods	Inference time [s]
SR	121.56±0.12
kNN-UE (K=8)	128.98±0.11
kNN-UE (K=16)	128.97±0.13
kNN-UE (K=32)	128.54±0.16
kNN-UE (K=64)	129.16±0.16
kNN-UE (K=128)	128.39±0.20

Table 14: Inference time [s] on IMDb test set when changing K in kNN-UE.

H Inference Time Results When Changing -K

To estimate whether the inference time changes significantly when changing Top-K in kNN search, we investigated the inference time when changing K on the IMDb test set. Table 14 shows that the inference time remains almost the same when changing K in the range of 8 to 128.

I Each Result of Product Quantization, Clustering, and Dimension Reduction

I.1 Product Quantization

(PQ) (Jégou et al., 2011) is a data compression technique based on vector quantization. In PQ, a D-dimensional representation is divided into N_{sub} subvectors and quantized by performing k-means clustering on the vectors in each subspace. Vector quantization can significantly reduce the amount of memory occupied by vectors.[13] In addition, by calculating the distance between compressed PQ codes, we can efficiently calculate the estimated value of the original Euclidean distance.

We evaluated UE performance and inference time when the number of clusters in the codebook was fixed at 32, and the number of subvectors was changed to $N_{\text{sub}} \in \{16, 32, 64\}$ (In Table 7 and 8, PQ was performed with $N_{\text{sub}} = 32$).

Table 15 shows the UE performance and inference time results in different N_{sub}. In ECE and E-AURC, there are almost no degradation in UE performance due to PQ. On the other hand, in MCE in ID setting, the UE performance consistently degrades. Furthermore, compared to kNN-UE among different N_{sub}, the larger N_{sub}, the better the UE performance tends to improve, but the inference time increases.

The larger N_{sub} is, the more time is required for inference but the UE performance improves. We assumed that these results are derived from the decrease in quantization error over the vector

[13]For example, raw datastore in kNN-UE is 636MB on OntoNotes 5.0 `bn`, but PQ reduces it to 10MB.

Methods	ECE (↓)	MCE (↓)	E-AURC (↓)	time [s]
	OntoNotes 5.0 `bn` (In-domain)			
SR	7.79±0.53	50.07±24.15	21.90±1.31	2.49±0.08
kNN-UE (w/o label)	3.37±0.71	33.15±3.65	17.63±0.66	4.94±0.10
kNN-UE	1.78±0.32	26.02±13.72	20.14±1.27	4.99±0.07
kNN-UE ($N_{\text{sub}} = 16$)	1.90±0.27	31.18±11.17	20.16±1.12	3.27±0.06
kNN-UE ($N_{\text{sub}} = 32$)	1.96±0.31	31.33±18.74	20.23±1.27	3.32±0.05
kNN-UE ($N_{\text{sub}} = 64$)	1.88±0.34	31.06±16.36	20.16±1.23	4.11±0.11
	OntoNotes 5.0 `nw` (Out-of-domain)			
SR	17.05±0.69	37.06±3.13	81.49±4.17	5.75±0.27
kNN-UE (w/o label)	8.78±0.62	24.91±1.81	70.10±4.03	10.36±0.21
kNN-UE	7.50±0.42	16.53±2.61	74.27±5.43	10.48±0.12
kNN-UE ($N_{\text{sub}} = 16$)	7.66±0.48	17.07±3.81	74.47±5.53	7.22±0.19
kNN-UE ($N_{\text{sub}} = 32$)	7.57±0.45	16.43±2.73	74.38±5.36	7.23±0.16
kNN-UE ($N_{\text{sub}} = 64$)	7.57±0.44	16.38±2.66	74.35±5.49	8.90±0.18

Table 15: ECE, MCE, E-AURC and inference time results about NER on OntoNotes 5.0 `bn` (In-domain) and OntoNotes 5.0 `nw` (Out-of-domain) for $\mathrm{mDeBERTaV3}_{\mathrm{BASE}}$ model when applied PQ in different N_{sub}.

with PQ with larger N_{sub} because each subvector is divided into smaller subspaces and the quantization is performed for each subspace. On the other hand, an increase in N_{sub} requires additional distance computations etc., then more inference time.

I.2 Clustering

The original kNN-LM uses an inverted file index (IVF) technique that speeds up the search by dividing the representation into N_{list} clusters by k-means and searching for neighbors based on N_{probe} centroids. In this study, we evaluate the UE performance and inference speed when the number of clusters $N_{\text{list}} = 100$. In this study, we evaluate the UE performance and inference speed when the number of clusters $N_{\text{list}} = 100$ and applying PQ with $N_{\text{sub}} = 32$ are fixed and the number of cluster centroids to search changes$N_{\text{probe}} \in \{8, 16, 32, 64\}$ (In Table 7 and 8, IVF was performed with $N_{\text{probe}} = 32$).

Table 16 shows the performance of UE when changing N_{probe} in ID and OOD settings using OntoNotes 5.0. In ECE, scores are slightly reduced for ID, but only slightly worse for OOD; MCE also shows degradation for ID but little for OOD, and even improves when $N_{\text{probe}} = 8$; E-AURC shows almost no change in scores when N_{probe} is changed for both ID and OOD. In terms of inference time, the larger N_{probe}, the longer it takes. We derive the improvement in MCE when increasing N_{probe} in ID setting from the fact that more clusters are targeted, making it possible to cover ground-truth nearest neighbor examples. On the other hand, the tendency of slight decrease when increasing N_{probe} in OOD setting may comes from the reliability of the vector, similar to the discussion in Section 6.3.

In addition, Taken together with the results in

Methods	ECE (↓)	MCE (↓)	E-AURC (↓)	time [s]
	OntoNotes 5.0 bn (In-domain)			
SR	7.79±0.53	50.07±24.15	21.90±1.31	2.49±0.08
kNN-UE (w/o label)	3.37±0.71	33.15±3.65	17.63±0.66	4.94±0.10
kNN-UE	1.78±0.32	26.02±13.72	20.14±1.27	4.99±0.07
kNN-UE ($N_{\text{probe}}=8$)	1.82±0.28	30.18±16.77	20.14±1.21	2.84±0.08
kNN-UE ($N_{\text{probe}}=16$)	1.86±0.25	29.48±16.91	20.13±1.21	3.11±0.03
kNN-UE ($N_{\text{probe}}=32$)	1.92±0.31	28.55±11.24	20.13±1.22	3.31±0.06
kNN-UE ($N_{\text{probe}}=64$)	1.83±0.28	27.00±9.43	20.14±1.21	3.71±0.06
	OntoNotes 5.0 nw (Out-of-domain)			
SR	17.05±0.69	37.06±3.13	81.49±4.17	5.75±0.27
kNN-UE (w/o label)	8.78±0.62	24.91±1.81	70.10±4.03	10.36±0.21
kNN-UE	7.50±0.42	16.53±2.61	74.27±5.43	10.48±0.12
kNN-UE ($N_{\text{probe}}=8$)	7.52±0.41	16.01±1.92	74.33±5.37	6.09±0.28
kNN-UE ($N_{\text{probe}}=16$)	7.56±0.36	16.93±3.38	74.31±5.39	6.65±0.17
kNN-UE ($N_{\text{probe}}=32$)	7.60±0.41	17.12±2.35	74.34±5.35	7.33±0.21
kNN-UE ($N_{\text{probe}}=64$)	7.53±0.40	17.28±2.45	74.33±5.37	7.89±0.12

Table 16: ECE, MCE, E-AURC and inference time results about NER on OntoNotes 5.0 bn (In-domain) and OntoNotes 5.0 nw (Out-of-domain) for $\text{mDeBERTaV3}_{\text{BASE}}$ model when applied IVF in different N_{probe}.

Table 7 in Section 7.3, we can see that the degradation of the UE performance can be mitigated with improvement latency when applying PQ and IVF with lower N_{probe}, compared to applying PQ, IVF and PCA simultaneously.

I.3 Dimension Reduction

In general, Transformer-based models such as PLM have high-dimensional token representations. In high-dimensional spaces, nearest neighbor search often suffer from the curse of dimensionality. To reduce this problem, we apply dimension reduction to kNN-UE similar to He et al. (2021a). In this study, we use Principal Component Analysis (PCA) as a dimension reduction algorithm to reduce the dimension of the datastore representations and the query representation D_{pca} (In Table 7 and 8, PCA was performed with $D_{\text{pca}}=128$). As shown in Table 17, the UE performance depends on the number of target dimensions, and the performance degrades when $D_{\text{pca}}=64$ or $D_{\text{pca}}=128$. On the other hand, the performance in $D_{\text{pca}}=256$ is almost the same as default kNN-UE. This suggests that excessive dimension reduction in distance computation to extract nearest examples by kNN search could have a negative impact on the UE performance.

J Distance Recomputation for kNN-UE

When using efficient kNN search techniques in Section 7.3, we use approximate distances to compute Eq. 4. Although we can get raw vectors by using the example indices obtained from approximate nearest neighbor search and compute accurate distance, in kNN-LM this has been shown to lead to performance gains and latency degradation (He et al., 2021a). We measure the UE performance and inference speed when PQ, clustering, and dimension reduction are applied simultaneously and re-computing accurate distances, reported in Table 18. These results show that the UE performance does not improve except for MCE in the ID setting, and the latency is about 5-7x slower when reading raw vectors from the datastore and re-computing distances. Moreover, these results suggest that exact distance computation for examples that are not actually nearest neighbors are not very effective in kNN-UE.

Methods	ECE (↓)	MCE (↓)	E-AURC (↓)	time [s]
	OntoNotes 5.0 bn (In-domain)			
SR	7.79±0.53	50.07±24.15	21.90±1.31	2.49±0.08
kNN-UE (w/o label)	3.37±0.71	33.15±3.65	17.63±0.66	4.94±0.10
kNN-UE	1.78±0.32	26.02±13.72	20.14±1.27	4.99±0.07
kNN-UE ($D_{\text{pca}}=64$)	1.89±0.37	31.01±14.35	20.06±1.25	3.24±0.08
kNN-UE ($D_{\text{pca}}=128$)	1.80±0.36	27.85±13.80	20.13±1.29	3.41±0.10
kNN-UE ($D_{\text{pca}}=256$)	1.80±0.40	26.23±12.61	20.13±1.28	3.85±0.06
	OntoNotes 5.0 nw (Out-of-domain)			
SR	17.05±0.69	37.06±3.13	81.49±4.17	5.75±0.27
kNN-UE (w/o label)	8.78±0.62	24.91±1.81	70.10±4.03	10.36±0.21
kNN-UE	7.50±0.42	16.53±2.61	74.27±5.43	10.48±0.12
kNN-UE ($D_{\text{pca}}=64$)	7.48±0.41	16.20±2.75	74.33±5.49	7.37±0.26
kNN-UE ($D_{\text{pca}}=128$)	7.54±0.45	16.42±2.73	74.30±5.44	7.75±0.24
kNN-UE ($D_{\text{pca}}=256$)	7.56±0.43	16.13±2.59	74.26±5.40	8.51±0.46

Table 17: ECE, MCE, E-AURC and inference time results about NER on OntoNotes 5.0 bn (In-domain) and OntoNotes 5.0 nw (Out-of-domain) for $\text{mDeBERTaV3}_{\text{BASE}}$ model when applied PCA in different D_{pca}.

Methods	ECE (↓)	MCE (↓)	E-AURC (↓)	time [s]
	OntoNotes 5.0 bn (In-domain)			
kNN-UE	1.78±0.32	26.02±13.72	20.14±1.27	4.99±0.07
kNN-UE (Approx.)	2.14±0.37	33.52±10.84	20.12±1.26	2.87±0.04
kNN-UE (Recomp.)	2.35±0.44	30.47±7.50	20.16±1.17	16.24±0.77
	OntoNotes 5.0 nw (Out-of-domain)			
kNN-UE	7.50±0.42	16.53±2.61	74.27±5.43	10.48±0.12
kNN-UE (Approx.)	8.08±0.53	24.03±5.46	74.50±5.42	6.20±0.20
kNN-UE (Recomp.)	8.30±0.51	25.67±5.26	74.58±5.53	34.22±0.78

Table 18: ECE, MCE, E-AURC and inference time results about NER on OntoNotes 5.0 bn (In-domain) and OntoNotes 5.0 nw (Out-of-domain) when applying distance recomputation in kNN-UE. "Approx." indicates using approximate distances, and "Recomp." indicates using exact distances by distance recomputation. Both "Approx." and "Recomp." are applied PQ with $N_{\text{sub}}=32$, clustering with $N_{\text{probe}}=32$ and dimension reduction with $D_{\text{pca}}=128$.

K Licenses of Datasets, Tools and Models

Datasets The IMDb movie dataset can be used for research purposes as described in https://developer.imdb.com/non-commercial-datasets/. Yelp dataset can be used for academic purposes as described in https://s3-media0.fl.yelpcdn.com/assets/srv0/engineering_pages/f64cb2d3efcc/assets/vendor/Dataset_User_

Agreement.pdf. The MNLI dataset is licensed for research purposes as described in Williams et al. (2018). The SNLI dataset can be used for research purposes as described in https://nlp.stanford.edu/projects/snli/. OntoNotes 5.0 dataset can be used for research purposes as described in https://catalog.ldc.upenn.edu/LDC2013T19.

Tools `transformers` is licensed by Apache-2.0. `faiss` is MIT-licensed.

Models $\mathrm{DeBERTaV3}_{\mathrm{BASE}}$ and $\mathrm{mDeBERTaV3}_{\mathrm{BASE}}$ from Huggingface model checkpoints are MIT-licensed.