**kNN-CM: A Non-parametric Inference-Phase Adaptation of Parametric Text Classifiers**

Rishabh Bhardwaj£* Yingting Liα* Navonil Majumder£
Bo Chengα Soujanya Poria£

£ Singapore University of Technology and Design, Singapore
α Beijing University of Posts and Telecommunications

rishabh_bhardwaj@mymail.sutd.edu.sg, cindyyting@bupt.edu.cn
navonil_majumder@sutd.edu.sg, chengbo@bupt.edu.cn
sporia@sutd.edu.sg

**Abstract**

Semi-parametric models exhibit the properties of both parametric and non-parametric modeling and have been shown to be effective in the next-word prediction language modeling task. However, there is a lack of studies on the text-discriminating properties of such models. We propose an inference-phase approach—\(k\)-Nearest Neighbor Classification Model (\(k\)NN-CM)—that enhances the capacity of a pre-trained parametric text classifier by incorporating a simple neighborhood search through the representation space of (memorized) training samples. The final class prediction of \(k\)NN-CM is based on the convex combination of probabilities obtained from \(k\)NN search and prediction of the classifier. Our experiments show consistent performance improvements on eight SuperGLUE tasks, three adversarial natural language inference (ANLI) datasets, 11 question-answering (QA) datasets, and two sentiment classification datasets. The source code of the proposed approach is available at [https://github.com/Bhardwaj-Rishabh/kNN-CM](https://github.com/Bhardwaj-Rishabh/kNN-CM).

**1 Introduction**

The recent advancements in Natural Language Processing (NLP) have largely been attributed to the learning of contextual representations of text from large language models acting as a backbone. Most of these language models, such as versions of BERT (Devlin et al., 2018), GPT (Radford et al., 2018), T5 (Raffel et al., 2020), are parametric, i.e., they encode information required to solve a task purely in its parameters.

A parametric model, irrespective of the size of the dataset, assumes that the output variables are dependent on the input variables through a predefined class of functions. The exact function is ascertained by learning its fixed set of parameters. For instance, a linear regression model fits a set of parameters in a function that defines a supposedly linear relationship between (independent) input variables and (dependent) output variables. As a complex composition of many linear regressions, many neural architectures, such as Transformer-based models (Vaswani et al., 2017), can thus be classified as purely parametric.

However, there has been little research on the utility of non-parametric models for NLP. In contrast to parametric models which need a predefined function, such as a linear model, non-parametric models seek training data to help define the function form itself. Thus, they provide flexibility to fit a wide range of possible shapes of ground truth function. A widely known non-parametric model is the \(k\)-nearest neighbor (\(k\)NN) where inference on test samples are drawn from the information provided by the neighborhood formed by train set samples (Fix and Hodges, 1989).

A \(k\)NN model provides memorization capabilities and captures rare patterns from the training set that otherwise are ignored by a parametric model (as studied by Khandelwal et al. (2019)). Language models (LMs) with non-parametric properties have shown impressive gains in next-word prediction tasks (Yogatama et al., 2021; Bhardwaj et al., 2022; He et al., 2021; Khandelwal et al., 2019). Additionally, these models do not need explicit parameter learning via optimization, thus cutting the model training time completely—the lack of such characteristics in a purely parametric model motivates the proposed approach.

This work explores the importance of querying neighbors to solve classification tasks in the text domain. We hypothesize that underlying language model representations have a high-dimensional spatial proximity relation between input instances which can be leveraged to enhance prediction performance—beyond the capacity of the classifier. Hence, we propose a semi-parametric model \(k\)NN-CM (\(k\) Nearest Neighbor Classification Model).
which constitutes a parametric classifier and a non-parametric memory (i.e. datastore) for neighborhood retrieval. In reinforcement learning (RL), classifiers are often employed as tools to aid policy learning or state representation. They can help estimate the quality of different actions or policies, and classify text into different categories, thus possessing high utility for the recent paradigm shifts in generative models (von Werra et al., 2020).

**Contributions.** We propose an inference phase semi-parametric modeling approach $k$NN-CM that enhances the capacity of a given parametric classifier model by incorporating an external memory datastore. In the inference phase, $k$NN-CM performs a $k$-neighborhood search through the datastore and merges the neighborhood predictions with the prediction obtained from the parametric classifier. Since the expansion of the CM to $k$NN-CM happens in the inference phase, it allows one to enhance the capacity of most of the existing pre-trained neural classifiers. By performing an extensive set of experiments, we demonstrate the importance of neighborhood search through the memorized samples on eight SuperGLUE tasks, three NLI datasets, 11 QA tasks, and two aspect-based sentiment classification tasks. We also show how the semi-parametric method can still outperform CM in out-of-domain scenarios. Furthermore, we test $k$NN-CM by tasking it under various cases of domain adaptation. Since $k$NN-CM introduces prediction latency compared with CM, we demonstrate how one can employ an entropy-based divergence measure to filter out the samples that use $k$NN retrieval facility. Additionally, we illustrate the importance of memorization in the low-resource scenario. In the end, we point out potential extensions of the proposed approach in conversation modeling and continual learning.

2 Related Work

**Computer vision.** For the image captioning task, Karpathy and Fei-Fei (2015); Devlin et al. (2015) proposed nearest neighbor baselines where they assign an unseen sample the (consensus of) captions of the training set images closest to it. Wang et al. (2019b) studied the utility of (transformed) neighborhood while performing few-shot object classification tasks. $k$NN has also been used to analyze learned image representations (Wallace and Hariharan, 2020) as well as to classify images (Zhang et al., 2023). For instance, Wu et al. (2018) performed image classification without class supervision but considering every instance as a class.

**Recommendation systems.** For session-based recommendation (based on current user interaction with the system), Kamehkhosh et al. (2017); Jannach and Ludewig (2017) showed that a neighborhood-based model output performs a GRU-based neural model on the studied tasks. $k$NN has been widely popular in collaborative filtering (memory-based) where the recommendation is done by the user or item neighborhood search (Su and Khoshgoftaar, 2009; Sarwar et al., 2001).

**Language models.** The advantage of querying nearest neighbors from a set of pre-trained LM’s representations of the training set (datastore) was first observed by Khandelwal et al. (2019) by proposing $k$NN-LM. This is followed by several works such as improving retrieval speed (He et al., 2021), $k$NN-LM adaptation (Bhardwaj et al., 2022), adaptive interpolation (Drozdov et al., 2022; Yogatama et al., 2021), and masked language modeling (Min et al., 2022).

To the best of our knowledge, this work is the first attempt to extensively study the importance of neighborhood search for text classification to enhance the capacity of parametric classifiers in the inference phase. Figure 1 gives an illustrative to motivate the idea.

3 Methodology

For a given domain, we obtain training data $S:=\{(x_1, y_1), ..., (x_N, y_N)\}$ where $x_i$ denotes input text in instance space $\mathcal{X}$ and $y_i$ is the label of its class defined in label space $\mathcal{Y}$. A learning system, a classifier, comes up with a prediction rule (hypothesis) $h$ that maps the input from $\mathcal{X}$ to a probability distribution over class labels in $\mathcal{Y}$.
Classifier (CM). Without the loss of generality, we consider a CM constitutes a backbone large language model and a trainable classification head. During the pre-training phase, the language model is assumed to have learned semantic high-dimensional text representations that can be tuned to solve a given downstream task. The input to CM is a sequence of tokens \( x = \{v_1, \ldots, v_n\} \) and output is a probability distribution over the class labels. CM is tasked to approximate the ground truth input-output mapping by learning parameters of a predefined function, i.e., the neural connections within CM. We denote the task-specific trained parametric predictor by \( h_{CM} \).

**kNN.** We use a well-known algorithm \( k \)-nearest neighbors for non-parametric modeling (Fix and Hodges, 1989). Using training samples of a task, we construct a datastore \( D = \{v(x_i), y_i\}_{i=1}^N \), where \( v(x) \) denotes the high-dimensional vector embeddings of text \( x \) obtained from the classifier model. For a given classification task, an unseen test sample \( \hat{x} \) is classified based on the nearest neighborhood training samples \( \{x_1, \ldots, x_N\} \). Let \( \arg\min_k \) denote the index of \( k \) training samples that return least distance values from \( \hat{x} \),

\[
K := \arg\min_k \frac{1}{N} \sum_{i \in [N]} d(v(\hat{x}), v(x_i)) 
\]

\[ p(y \in \mathcal{Y}) := \frac{\sum_{i \in K} 1[y_i = y]}{k} \tag{1} \]

where \( d(\cdot) \) denotes the distance function between \( v(\hat{x}) \) and \( v(x_i) \), \( y \) denotes the label of \( x_i \). Similar to Khandelwal et al. (2019); Bhardwaj et al. (2022), we use euclidean distance for \( d(\cdot) \). Hence, we obtain a non-parametric hypothesis \( h_{kNN} \). We define the semi-parametric classifier model \( kNN-CM \) as a linear combination of the two probability distributions with coefficient \( \lambda \in [0, 1] \)

\[
h_f := \lambda h_{kNN} + (1 - \lambda) h_{CM}. \tag{2} \]

There are several aspects of this formulation:

- While performing kNN search, parametric classifier parameters are kept frozen.
- Strong dependence of \( h_{kNN} \) on \( h_{CM} \): Unlike commonly used ensemble methods where the underlying classifiers undergo independent training and inference, the errors made by nearest neighbor classifier highly depends on the effectiveness of its search space (datastore), which is defined by the vector representations of text provided by CM.
  - Explicit control over a model capacity: Integrating \( kNN \) with CM provides explicit control over the model’s capacity. For instance, a change in the \( k \) value changes the model’s bias and variance as shown in the non-parametric estimator’s study by Geman et al. (1992). Changing a model’s bias-variance characteristics directly affects the model’s capacity to fit a wider class of functions\(^2\).
  - We hypothesize that neighborhood search is important when the classifier is confused between classes and prone to do mistakes around the decision boundary. We quantify this aspect and call a model more confused if the classifier’s output probabilities resemble a uniform distribution. Thus, one can choose between CM and \( kNN-CM \) depending on the unseen sample under testing. We study this aspect in detail in Section 5.

Next, we define the error made by \( h_f \) over \( m \) test samples

\[
\epsilon := \frac{\left| \left\{ i \in [m] : \arg\max_{j \in \mathcal{Y}} [h_f^j(x_i)] \neq y_i \right\} \right|}{m} \tag{3} 
\]

where \( h_f^j \) is probability assigned by the hypothesis to class \( j \), \( |\cdot| \) is the cardinality of the set and \( |m| = \{1, \ldots, m\} \). Note that \( 1 - \epsilon \) denotes the accuracy of the semi-parametric classifier.

**Time and Space Complexity.** Similarity search can be computationally expensive and introduce high memory requirements for each task, thus we use Faiss (Johnson et al., 2019)—an efficient similarity search algorithm and clustering (indexing) algorithm of high dimensional data. The clustering of similar vectors in high dimensions obviates the need to search through the whole training set (datastore). For small-scale datasets, we use IndexFlatL2 which queries through all the vectors in the datastore. The complexity is thus \( O(n) \) where \( n \) is the number of elements in the datastore. For large-scale datastores, we use IndexFlatIVF to first

\(^2\)Broadly, a model with \( k = N \) will learn fewer patterns from data while a model with \( k = 1 \) can learn as many patterns as \( N \), where \( N \) is the number of training samples.
cluster the vectors in datastore and then perform an NN search in each cluster. The time complexity of this method is $O(n_c m)$ where $n_c$ and $m$ denote the number of clusters and the average number of elements in clusters, respectively. The space complexity of IndexFlatL2 is $O(nd_s)$ and IndexFlatIVF is $O(nd_s + md_s)$, where $d_s$ denotes the dimensionality of the vector in the datastore. Contrary to non-parametric, the time and space complexity of a parametric model, such as CM, is predefined and does not vary with the number of train samples.

4 Experiments

4.1 Toy dataset

We hypothesize that a CM is prone to miss neighborhood cues from the training data. To test this, we set up a toy experiment on a neural network comprised of one hidden layer of 100 nodes activated with ReLU. To test the capacity of this network, we synthesize a dataset by randomly initializing 20 cluster centers $\{c_i : c_i \sim N(0, 1.5), i \in 1 \ldots 20\}$, each of which constitutes 20 points $\{c_{ij} : c_{ij} = c_i + p_j; p_j \sim N(0, 1), j \in 1 \ldots 20\}$, where cluster center and $p_j$ are independently sampled for each of their dimensions. All the data points lie in the space of $\mathbb{R}^{100}$. We randomly split the clusters into four classes. Figure 2 shows the 2-dimensional t-SNE plot of the generated data with samples shown in the same color belonging to the same class and vice versa. Circles represent samples used to learn the network parameters, black dots denote the correctly classified test cases and the red squares denote the test samples incorrectly classified by the network. Red squares provide evidence for the hypothesis, i.e., while the model is able to identify correct clusters of several test cases, it still fails to capture the nuance of the neighborhood precisely.

4.2 NLP datasets

We base our main experiments on the SuperGLUE benchmark and a large variety of existing NLP datasets to solve NLI, Question Answering (QA), and Sentiment Classification.

SuperGLUE (Wang et al., 2019a). It is a benchmark dataset to evaluate a model on its language understanding capabilities. BoolQ (Clark et al., 2019) is a QA task where a yes/no question is asked on a short passage. CB (De Marneffe et al., 2019) is a textual entailment task where given a premise, the model is asked to predict how committed the author is toward the truth of the (clause) hypothesis. COPA (Roemmele et al., 2011) is a causal reasoning task to identify the cause or effect of a premise from the set of given choices. MultiRC (Khashabi et al., 2018) is a multi-choice QA task where a question is asked about a context paragraph and the answer choices are provided. ReCoRD (Zhang et al., 2018) is a multi-choice QA task where, from a passage, a masked-out entity is to be predicted from a set of entities. RTE (Haim et al., 2006) is another textual entailment dataset with two classes, entailment and not entailment. WiC (Pilehvar and Camacho-Collados, 2018) is a task of word sense disambiguation. Given two texts and a word appearing in both sentences, the task is to determine if the word is used in the same sense in both sentences. WSC (Levesque et al., 2012) is a conference resolution task where an example consists of a pronoun and a list of noun phrases, the task is to identify the correct pronoun referent.

BoolQ, COPA, COPA, WiC, WSC, and RTE are binary classification tasks, CB three-class classification, MultiRC and ReCoRD are cast as binary class classification where the correct choice (or entity) is labeled 1 and incorrect is labeled as 0.

ANLI (Nie et al., 2020). Adversarial Natural Language Inference is a large-scale benchmark NLI dataset constructed by an adversarial human-model-in-loop. The dataset is subdivided into three datasets A1, A2, and A3 with increasing task difficulty. ANLI aims to solve a textual entailment task where given a premise, the model is asked to predict if a hypothesis entails, contradicts, or is neutral to the premise. We use ANLI to represent combination of A1, A2, and A3.
**Question Answering.** For QA tasks, we experiment on ten datasets: QASC (Question Answering via Sentence Composition) (Khot et al., 2020) is a fact retrieval from a large corpus to answer a question given eight choices, only one of which is correct. PIQA (Physical IQA) (Bisk et al., 2020) tests physical knowledge of language models by asking them to select the correct choice from the given two. SIQA (Social IQA) (Sap et al., 2019) is a common sense reasoning dataset for the context of social situations. Given a social situation and three choices to select from, the task is to select the correct choice. CQA (CommonsenseQA) (Talmor et al., 2019) is a commonsense QA based on ConceptNet knowledge (Speer et al., 2017). For a question, the task is to choose one of five given choices. CQA-2 (CommonsenseQA 2.0) (Talmor et al., 2021) is another recent commonsense QA dataset constructed with model-in-loop approach. It consists of commonsense questions from various categories of reasons, with the answer being yes or no. SWAG and (H-SWAG) (Zellers et al., 2018) are datasets for grounded inference. Given an incomplete event description, the task is to find the correct ending from a set of four choices. CosmosQA (Huang et al., 2019) is a dataset for commonsense-based reading comprehension. The task is to identify a correct choice from the given four for the question asked about a paragraph. CICERO v1, v2 (Ghosal et al., 2022b; Shen et al., 2022) are dialogue QA dedicated datasets. Given a question about a given utterance taken from a dialogue, the task is to choose the correct answer from the choices.

**Aspect-Based Sentiment Classification.** We also compare the proposed approach on two aspect-based sentiment classification datasets—Laptop and Restaurant. The datasets are a set of restaurant and laptop reviews obtained from Pontiki et al. (2015, 2016). We convert the given review in the form $w_1 \ w_2 \ldots <\text{aspect term}> \ldots w_n$, where $<$ encloses the aspect term for which the sentiment (positive/negative/neutral) is to be predicted.

### 4.3 Experimental Setup

The ANN-CM is an inference phase approach that does not require task-specific CM parameter tuning. We either train a classifier or utilize existing pre-trained task-specific classifiers to obtain a baseline CM on a given task.

**CM Setup.** For all the tasks in the SuperGLUE benchmark, we utilize RoBERTa-base ([Liu et al., 2019]) as the backbone language model. Following the success of parameter-efficient adapters (Houlsby et al., 2019), and their competitive performance with full-mode fine-tuning (Liu et al., 2022; Hou et al., 2022; Bhardwaj et al., 2022), we obtain a task-specific classifier (CM) by training adapter modules inserted in between LM layers and attaching a classification module (head) on top of the LM. All the tasks are formulated as a classification problem. We follow a similar setup for language inference tasks (ANLI) and sentiment analysis tasks. For QA datasets, we use the DeBERTa-large (He et al., 2020) based classifier. Following TEAM (Ghosal et al., 2022a) which has shown a better than baseline performance on numerous QA tasks, we formulate all the multi-choice QA tasks as binary classification where the correct choices are labeled as 1 and incorrect choices are labeled as 0. Therefore, in the training phase, the classifier model aims to minimize the binary-cross entropy objective function. During the inference phase, we select the choice with the maximum class 1 probability score. Since our approach improves the model performance in the inference phase, we liberate ourselves from classifier training by downloading the model checkpoints generously provided by Ghosal et al. (2022a). The classification head uses $<$s> from RoBERTa and [CLS] from DeBERTa (generally used as classification tokens).

**kNN Setup.** For each task under study, we use the task-specific trained CM obtained via the method described above and construct a datastore using the train set samples. We obtain hidden representations of each sample by performing one forward pass through CM. For fast neighbor search and making the datastore memory-efficient, the obtained vectors are indexed using Faiss.

**Hyperparameters.** Since the approach is applicable to the inference phase, the primary set of hyperparameters comes from $k$NN search. For each task, we find the best interpolation parameter $\lambda$ (Equation (2)) and the optimal number of neighbors to search $k$ using the validation set. $\lambda \in \{0.001, 0.01, 0.1, 0.2, \ldots, 0.8, 0.9, 0.99, 0.999\}$ and $k \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$.

---

3We utilize the data collected by Zhou et al. (2021)

4Trainable parameters are $\approx 1.2\%$ of classifier parameters.

5Datasets downloaded from https://huggingface.co/datasets/super_glue
Table 1: Performance comparison of kNN-CM vs. CM on the SuperGLUE development set. The results are reported on the validation set.

<table>
<thead>
<tr>
<th>Task</th>
<th>Acc. F1</th>
<th>kNN-CM Acc. F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CB</td>
<td>91.07</td>
<td>92.86(11.96%) 92.37(14.46%)</td>
</tr>
<tr>
<td>COPA</td>
<td>56.00</td>
<td>64.00(11.29%) 63.87(14.38%)</td>
</tr>
<tr>
<td>WSC</td>
<td>63.46</td>
<td>63.40(10.05%) 47.41(17.04%)</td>
</tr>
<tr>
<td>RTE</td>
<td>77.98</td>
<td>79.42(12.85%) 79.18(12.04%)</td>
</tr>
<tr>
<td>WiC</td>
<td>69.12</td>
<td>69.28(10.23%) 69.28(10.29%)</td>
</tr>
<tr>
<td>BoolQ</td>
<td>80.49</td>
<td>81.16(10.83%) 79.86(11.57%)</td>
</tr>
<tr>
<td>MultiRC</td>
<td>66.56</td>
<td>70.40(15.77%) 69.26(14.04%)</td>
</tr>
<tr>
<td>ReCoRD</td>
<td>61.17</td>
<td>62.05(14.43%) 62.72(14.45%)</td>
</tr>
</tbody>
</table>

Table 2: Results on the test set of NLI tasks. ANLI is a combined dataset A1, A2, A3.

Table 3: kNN-CM vs. CM on QA datasets. Bin. and Inst. denote binary and instance classification accuracy.

<table>
<thead>
<tr>
<th>Task</th>
<th>CM Bin.</th>
<th>CM Inst.</th>
<th>kNN-CM Bin.</th>
<th>kNN-CM Inst.</th>
</tr>
</thead>
<tbody>
<tr>
<td>QASC</td>
<td>91.27</td>
<td>75.05</td>
<td>91.10</td>
<td>75.92(11.15%)</td>
</tr>
<tr>
<td>QASC-IR</td>
<td>95.39</td>
<td>88.01</td>
<td>95.40</td>
<td>88.44(10.49%)</td>
</tr>
<tr>
<td>PIQA</td>
<td>73.59</td>
<td>86.45</td>
<td>73.91</td>
<td>86.94(10.57%)</td>
</tr>
<tr>
<td>SiQA</td>
<td>81.01</td>
<td>80.71</td>
<td>80.98</td>
<td>81.27(10.69%)</td>
</tr>
<tr>
<td>CQA</td>
<td>88.06</td>
<td>82.80</td>
<td>88.12</td>
<td>83.29(10.59%)</td>
</tr>
<tr>
<td>CQA-2</td>
<td>54.37</td>
<td>57.97</td>
<td>61.20</td>
<td>60.84(14.95%)</td>
</tr>
<tr>
<td>SWAG</td>
<td>91.46</td>
<td>93.14</td>
<td>91.47</td>
<td>93.24(10.11%)</td>
</tr>
<tr>
<td>CosmosQA</td>
<td>87.63</td>
<td>86.30</td>
<td>87.86</td>
<td>86.77(10.54%)</td>
</tr>
<tr>
<td>H-SWAG</td>
<td>94.44</td>
<td>96.15</td>
<td>94.74</td>
<td>96.24(10.99%)</td>
</tr>
<tr>
<td>CICERO-v1</td>
<td>88.53</td>
<td>83.55</td>
<td>88.50</td>
<td>83.70(10.18%)</td>
</tr>
<tr>
<td>CICERO-v2</td>
<td>87.62</td>
<td>90.34</td>
<td>87.67</td>
<td>90.56(10.24%)</td>
</tr>
</tbody>
</table>

5 Results and Discussion

Table 1 shows results on SuperGLUE datasets. We observe kNN to correct predictions of CM in textual entailment tasks such as CB and RTE. The assistance of neighbors from the hidden space representations also shows a huge improvement (by ≈7%) to resolve the ambiguity in pronoun in WSC. However, the improvement in WiC is comparably less (≈0.3%). After investigation, we found that CM and kNN share the same set of samples where they make erroneous predictions. The conclusions are made on relatively low improvements on BoolQ when compared with MultiRC and ReCoRD. Amongst all, we observe a huge improvement for over 14% in COPA. We notice that kNN alone can surpass the baseline COPA accuracy by over three points. While a combination of both gives a boost of over eight points in both performance matrices.

Table 2 shows the improvement due to kNN involvement during predictions. We find the neighborhood search to help more as the task complexity increases, thus the observed improvement for A3 is more than the other adversarial datasets. More improvement in F1 score indicates that neighborhood search is less impacted by the class imbalance when compared with the CM-only setting. The best k identified for ANLI tasks is between 1-4 with high λ (≈0.99), thus, reflecting the information provided by the closest few neighborhoods is important and sufficient.

In QA tasks (Table 3), the observed improvement is relatively lower when compared with SuperGLUE and ANLI. After investigation, we observed the prediction that the predictions made by CM and kNN are similar. This indicates an effective clustering by the (DeBERTa-large) CM to perform the binary classification task. For instance, on SIQA, the instance accuracy of kNN is 80.45% while the CM performance is 80.98% with errors made on a similar set of samples.

Table 4 shows the results on sentiment analysis tasks. Similar to WiC, the reasons for poor performance on Restaurant were found to be the same set of erroneous predictions, and thus no explicit probability corrections were made by the neighbor samples. In contrast to this, we observed good performance on Laptop because the nearest neighbors help boost recall and precision.

Out-Of-Domain Performance. We retrieve SuperGLUE diagnostic datasets AX$_d$ and AX$_g$ (test only) and perform ANLI out-of-domain evaluations. Table 5 shows that the neighbor search on ANLI datastore not only improves on in-domain datasets but also shows over 12% F1 improvement.
Table 5: ANLI out of domain evaluation on AX₆, AX₇.

<table>
<thead>
<tr>
<th>Metric</th>
<th>CM</th>
<th>kNN-CM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc.</td>
<td>41.07</td>
<td>50.00</td>
</tr>
<tr>
<td>F1</td>
<td>32.19</td>
<td>34.73</td>
</tr>
</tbody>
</table>

Table 6: ANLI→CB domain adaptation without CM fine-tuning, by adding domain-specific datastore. $kNN_c$-CMₐ is a $kNN$-only classifier that constructs a datastore from RoBERTa-base LM. CMₐ denotes an ANLI classifier. $kNN$ subscripts ‘a’ and ‘c’ indicate the datastore is constructed using ANLI or/and CB training set.

Table 7: ANLI→CB domain adaptation with CM fine-tuning, by adding domain-specific datastore. $kNN_c$-CMₐ denotes an ANLI classifier. $kNN$ subscripts ‘a’ and ‘c’ indicate the datastore is constructed using ANLI or/and CB training set.

Table 8: ANLI→CB domain adaptation without CM fine-tuning, by adding domain-specific datastore. $kNN_c$-CMₐ is a $kNN$-only classifier that constructs a datastore from RoBERTa-base LM. CMₐ denotes an ANLI classifier. $kNN$ subscripts ‘a’ and ‘c’ indicate the datastore is constructed using ANLI or/and CB training set.

Domain Adaptation without Classifier Tuning. We carry out $kNN$-CM domain adaptation of ANLI to CB without explicitly fine-tuning the classifier but including the domain-specific datastore. In Table 6, we observe the datastore from CB domain is important to help boost the performance significantly. Even on an untrained classifier, merely including domain-specific datastore constructed on purely pre-trained LM (CMₐ) and classifying using $kNN$-only, gives 50% accuracy. The best-performing model is $kNN_c$-CMₐ (a CB datastore constructed on ANLI classifier) with an accuracy of around 75% and F1 score of over 53%. Merging available ANLI datastore with CB datastore, however, tends to reduce the performance. We posit the reason is a very small fraction of neighbors belonging to CB as compared to ANLI ($\approx 0.15\%$).³ Rescoring methods can help adapt the existing datastore to other domains (Bhardwaj et al., 2022).

Filtering Samples for $k$-NN Search. As discussed in section 3, we hypothesize the neighborhood search is important for CM to make important decisions around the decision boundaries which leads to model confusion. We assume the model to be more in need of external aid when the CM predictions are close to the uniform distribution over labels. Thus, we define a neighbor requirement score $r$ for a given text $x$ as the normalized KL-divergence of CM prediction with respect to a discrete uniform distribution over classes:

$$r(x) := \frac{\text{KL}(h_{CM}(x) || U_Y)}{\log(|Y|)}.$$

For a given input $x$, we redefine the predictor:

$$h_f(x) := \begin{cases} \lambda h_{kNN}(x) + (1-\lambda) h_{CM}(x), & \text{if } r(x) \leq \tau \\ h_{CM}(x), & \text{otherwise} \end{cases}$$

where $\lambda \in [0, 1]$, $|Y|$ (cardinality of the label set) denote the number labels, $U_Y$ denotes the uniform distribution over the labels in the set $Y$. $h_{CM}^i(x)$ is classifier $i$’s probability over label $i$, $\tau$ defines a threshold on divergence value below which $kNN$ will be involved in model predictions. In Figure 3, we observe ANLI accuracy to converge at $\tau = 0.7$. Thus, using entropy-based measures, one can filter samples for $kNN$ to reduce inference time.

Layer Importance for Retrieval. Following Khandelwal et al. (2019); Bhardwaj et al. (2022), we create datastore on representations obtained from different layers of ANLI-based classifier and perform hyperparameter search ($k$ and $\lambda$) on ANLI development set. Figure 4 shows the layer-wise test set performance increases when we go deeper in the network. For ANLI, we find the best-performing neighborhood representations to be layer normalization in the pre-final layer. In our initial experiments on the SuperGLUE benchmark, on average, we found the final layer to be the best performing.

³We use hyperparameters $k$ and $\lambda$, obtained from ANLI dev set, across the settings in Table 6.
Figure 4: Impact of different layers on ANLI. The horizontal axis denotes layer representations considered for datastore construction and the vertical axis denotes the test accuracy/F1.

$k$NN-CM Time Overhead. Being a semi-parametric model, the $k$NN search space tends to linearly increase with the increase in the number of training samples to memorize. Thus, we study the time-overhead, i.e., added inference-time latency due to the neighborhood search. Without the loss of generality, we base our analysis on the ANLI dataset. On the CPU \(^8\), the per-sample CM inference speech is \(\approx 72\) ms, and for the $k$NN retrieval \(^9\), it is \(\approx 29\) ms. On the GPU \(^{10}\), the per-sample in the CM stage is \(\approx 9\) ms, and for the $k$NN stage is \(\approx 2\) ms. Thus, a flat $k$NN search increases inference time by around 40% on CPU and 20% on GPU.

Utterance Classification. We also trained classifiers on datasets for emotion recognition in conversation. Given an utterance from a conversation with an appended set of eight utterances preceding it, we aim to classify it in one of the emotion classes. Our experiments on MELD (Poria et al., 2018), DailyDialogue (Li et al., 2017), and IEMOCAP (Busso et al., 2008) shows very insignificant improvements in Accuracy and F1 scores when the model is equipped with $k$NN search. We leave the precise semi-parametric modeling for utterance classification as future work.

Neighbors Weighting. We compare the frequency-based probability computations in Equation 1 with weighting neighbors with their distances from the query, thus

\[
p(y \in Y) := \sum_{i \in K} \mathbb{I}[y_i = y] \exp \left( \frac{-d(v(\hat{x}), v(x_i)))}{\beta}\right)
\]

In our initial experiments on QASC dataset with \(\beta \in \{2, 10, 20\}\), we found the validation set performance to be 75.81%, 75.92%, 75.92% all of which are higher than baseline CM but lower than frequency-based computations. We posit there is no generic weighting scheme that works for all the tasks, hence we leave methods to find a task-adaptive neighborhood weighting for classification for future work.

6 Conclusion

In this work, we presented $k$NN-CM, a semi-parametric paradigm that augments the inference

---

\(^8\)AMD Ryzen Threadripper 3960X 24-Core.
\(^9\)We implement IndexFlatL2 search.
\(^{10}\)NVIDIA RTX A6000.

<table>
<thead>
<tr>
<th>Task</th>
<th>CM Acc. F1</th>
<th>$k$NN-CM Acc. F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>20%</td>
<td>33.37 16.68</td>
<td>35.38 35.09</td>
</tr>
<tr>
<td>40%</td>
<td>33.37 16.68</td>
<td>33.37 31.18</td>
</tr>
<tr>
<td>60%</td>
<td>33.37 16.68</td>
<td>36.41 36.21</td>
</tr>
<tr>
<td>80%</td>
<td>46.47 44.90</td>
<td>48.09 47.43</td>
</tr>
<tr>
<td>100%</td>
<td>41.84 40.32</td>
<td>44.72 43.71</td>
</tr>
</tbody>
</table>

Table 7: Low-resource performance on ANLI.

**Low-Resource.** In Table 7, we observe that when data availability is reduced below 60%, the classification model performs worse equivalent to a random classification with uniform probabilities. When $k$NN is introduced, the performance of the $k$NN-CM model tends to be better than assigning the random labels to the test instances. It shows that, even in low resource cases, there are clustered vectors that $k$NN can exploit to boost the classification performance, while large parametric classifiers fail to capture proximity relations. This to some extent provides evidence to the hypothesis of Khan delwal et al. (2019), i.e., learning the similarity between texts is easier than predicting the next word, given our problem is reformulated as a label (text) generation. On the other hand, when the amount of training data is greater than 80% (of full-set), our baseline performed well and $k$NN adds further improved by nearly 4%-8%. Thus, irrespective of the baseline performance, $k$NN tends to maintain a better performance than random classification and it is evident that it supports the CM relentlessly even in the low-resource regime.
phase with neighborhood search through the training set. We studied the impact of adding non-parametric characteristics to a parametric classifier on 24 language understanding tasks. We further demonstrated the generalizability of \(k\)NN-CM by studying their out-of-domain and domain adaptation performance. We also showed its efficacy in low-resource scenarios where CM performance reduces dramatically and neighborhood search emerges as a savior. Toward the end, we leave a few important remarks on utterance classification and neighborhood weighting that carry the potential to motivate future research directions, which we elaborate on in the limitations section.

7 Limitations

We discuss the potential limitations of semi-parametric modeling and considerable future work:

- Non-parametric characteristics introduce challenges in the interpretability of the predictions.

- Since the function form highly depends on the size of the training set, the memory footprint grows linearly.

- Learning a good representation of the dataset is still a bottleneck task and predominantly relies on the parametric models. Thus, the performance and function form of non-parametric models depends on the effectiveness of the data representations.

- Since nearest neighbor computation requires pairwise similarity between test and samples in the train set, the inference time increases with the increase in the dimensionality of space and size of the train set. Several tools such as Faiss (Johnson et al., 2019) assist in a significant reduction of computational overhead with the trade-off in the performance of the model.

- One can compute \(k\)NN probabilities by using exponential of negative of distance (Khandelwal et al., 2019). However, simple averaging shows considerable improvements, and finding better probability computations is left for future work.

In the future, we see a huge potential of \(k\)NN’s in tackling the catastrophic forgetting in continual learning applications involving text. Another interesting area will be to propose methods that allow task-specific datastore representation tuning, more interestingly through backpropagation. Since the datastore size increases linearly with the number of training samples, scaling semi-parametric systems can be a challenging task. Thus, deploying such systems on edge devices with constrained computational capacity and memory is another interesting future research direction.

Acknowledgement

We thank the anonymous reviewers for their constructive feedback. This project is supported by the AcRF MoE Tier-2 grant (Project no. T2MOE2008, and Grantor reference no. MOE-T2EP20220017) titled: “CSK-NLP: Leveraging Commonsense Knowledge for NLP”, and the SRG grant id: T1SRIS19149 titled “An Affective Multimodal Dialogue System”. This work is also generously supported by BUPT Excellent Ph.D. Students Foundation CX2021229.

References


Haokun Liu, DEREK TAM, MOHAMMED MUQEEETH, JAY MOHTA, TENGHAO HUANG, MOHIT BANSAL, and COLIN RAFFEL. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. ArXiv, abs/2205.05638.


