# QUEST: Efficient Extreme Multi-Label Text Classification with Large Language Models on Commodity Hardware

**Chuang Zhou**[1], **Junnan Dong**[1], **Xiao Huang**[1,*],
**Zirui Liu**[2], **Kaixiong Zhou**[3], **Zhaozhuo Xu**[4]
[1]The Hong Kong Polytechnic University, China; [2]University of Minnesota, USA
[3]North Carolina State University, USA; [4]Stevens Institute of Technology, USA
[1]{chuang-qqzj.zhou, hanson.dong}@connect.polyu.hk; xiaohuang@comp.polyu.edu.hk
[2]zrliu@umn.edu; [3]kzhou22@ncsu.edu; [4]zxu79@stevens.edu

## Abstract

Extreme multi-label text classification (EMTC) involves predicting multiple labels from a vast pool of candidates based on a user's textual query. While traditional BERT-based methods have shown limited success, large language models (LLMs) have brought new possibilities. It is promising to leverage their remarkable comprehension ability to understand textual queries. However, implementing LLMs is non-trivial for two main reasons. Firstly, real-world EMTC datasets can be extremely large, with candidate product pairs reaching up to ten million in real-world scenarios, which poses significant challenges in data ingestion. Secondly, the large size of LLMs makes computation and memory demands prohibitive for EMTC applications. To this end, we propose QUEST, a **Qu**antized and **E**fficient Learning with **S**ampling **T**echnique. QUEST includes a tailored hash sampling module that reduces the data volume to one-fourth of its original size. Additionally, we perform compressive fine-tuning LLMs with only twenty thousand trainable parameters, largely reducing computational requirements. Extensive experiments demonstrate that QUEST outperforms existing methods while requiring fewer computational resources, unlocking efficient EMTC on commodity hardware such as a single Nvidia RTX 3090 GPU with 24 GB of memory.

## 1 Introduction

Extreme multi-label text classification (EMTC) has been widely applied in real-world scenarios, e.g., recommendation systems and social networks (Li et al., 2019; Ding et al., 2021), where each input query needs to be assigned multiple output labels from a very large set of candidate labels. As shown in figure 1, the input consists of text descriptions of purchased items, and the model is required to infer potential products from a vast list. Due to the complexity brought by its scale, EMTC has been extensively studied in natural language processing from various perspectives (Liu et al., 2017; Chang et al., 2021; Chalkidis et al., 2020). The existing standard method of EMTC is training a deep model to learn representations for input queries and labels. This process transforms tasks into a similarity search problem, where for each query, we search for its nearest neighbors in the label space. Early studies design linear models defined on TF-IDF feature vectors (Ramos et al., 2003). Recently, plenty of research has been dedicated to sentence embeddings powered by Bert-based models with million-scale parameters (Zhang et al., 2022). However, performance remains unsatisfactory due to its limited semantic understanding ability.

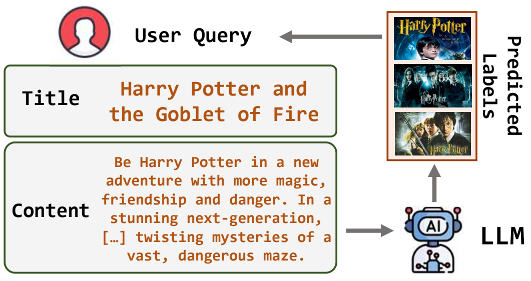

User Query
Title
Harry Potter and the Goblet of Fire
Content
Be Harry Potter in a new adventure with more magic, friendship and danger. In a stunning next-generation, […] twisting mysteries of a vast, dangerous maze.
LLM
Predicted Labels

Figure 1: A real example of directly applying LLMs for EMTC from the benchmark Amazon dataset.

With the emergence of large language models (LLMs), which have demonstrated impressive inference capabilities (Dong et al., 2024b; Chen et al., 2024; Li et al., 2019, 2023), we are motivated to incorporate LLMs into EMTC to better understand the semantics in both queries and labels. However, this task remains challenging for two reasons. Directly using LLMs to process extremely large training data is time-consuming, making it difficult to apply in real-world scenarios. Second, the extensive size of LLMs requires significant computational resources (Dong et al., 2024a), with a full llama-7B model consuming 28GB of memory. Moreover, the pace of our hardware enhancements

[*]Xiao Huang is the corresponding author.

is decelerating (Theis and Wong, 2017). Current GPU resources can handle lighter frameworks like Bag-of-words and Bert-based encoders, but fine-tuning a full LLM on a single machine is impossible. Furthermore, the average number of labels per query can reach up to 22.2 (Bhatia et al., 2016), further increasing the training pairs. Processing these extensive texts poses a heavy burden, necessitating the employment of multiple GPUs for over a week.

To this end, we present a novel **q**uantized and **e**fficient **l**earning framework with **s**ampling **t**echnique, i.e., QUEST, for extreme multi-label text classification, taking into account both *memory usage* and *computation resources*. We significantly reduce the search space of a large number of labels and facilitate efficient model training tailored for LLMs. (i) We introduce a novel hashing-based sampling module that selects more representative and diverse texts as training pairs for contrastive learning. This is inspired by the observation that textual descriptions of labels are highly clustered. We achieve promising performance by using only a small proportion of the original data. (ii) The model is optimized through a prompt-based pairwise contrastive training algorithm on the quantized LLM. This effectively reduces the number of trainable parameters and memory usage.

**Contributions:**

- We formally define the paradigm of efficient learning for EMTC with LLMs, addressing both *memory usage* and *computation resources*.
- We introduce the QUEST framework to achieve the selection of more representative textual data, thus boosting the training process through utilizing a customized diverse sampling technique.
- We enhance the adaptation of LLMs for EMTC tasks on commodity hardware with a prompt-based pairwise contrastive training algorithm on the quantized LLM, significantly reducing trainable parameters and memory usage.
- Extensive experiments demonstrate the superiority of QUEST in both efficiency and accuracy. QUEST outperforms all state-of-the-art methods within only 46.89 training hours on datasets containing over one million labels, using a single Nvidia RTX 3090 GPU with **24 GB** of memory.

## 2 Preliminaries

**Notations.** The dataset includes queries and labels, with each consisting of the title and content of the item description. In this paper, $X = \{(t_1, c_1), ..., (t_{|x|}, c_{|x|})\}$ is denoted as a collection of input documents, where each pair $(t_i, c_i)$ represents the title and content respectively. We denote $Y = \{y_1, ..., y_{|y|}\}$ as the set of labels, each of which is characterized by a piece of sentence. The original documents are split into the training dataset $X_{train}$ and the validation dataset $X_{test}$.

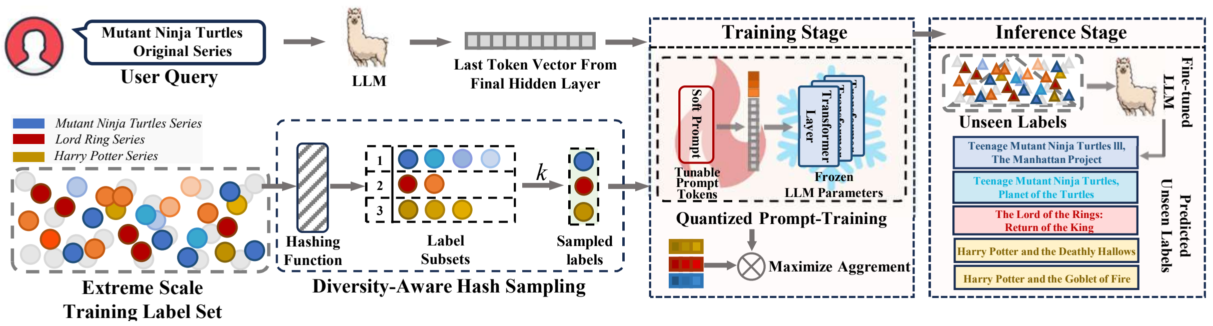
Different from ordinary classification tasks that only consider a limited number of labels (Zhou et al., 2023), the presence of millions of distinct choices makes using softmax classifiers difficult. Therefore, these issues are transformed into a nearest-neighbor search problem, where the model seeks to match the representations of queries with relevant tags. Another important aspect is its zero-shot capability. In reality, numerous newly released items are introduced to customers, making it impractical to have access to all potential labels during the training process. Providing accurate recommendations for unseen products holds great realistic significance.

## 3 Overall Framework

As mentioned in the introduction, this paper aims to achieve a more efficient and faster learning process. The approaches are mainly divided into two lines: increasing batch size and reducing parameters to improve **model efficiency**; and using diverse sampling to increase **data efficiency**. In this section, we will offer an overview of the motivation and intuitive explanations for each module.

### 3.1 Structural Text Representation

The first step is to obtain embeddings for both input queries and labels. In numerous instances, they are not merely a corpus but rather comprise various components, including titles and content. The current methods can be summarized in two lines: 1) generate embeddings of each component and merge piecewise vectors together (Ramos et al., 2003; Lee et al., 2019). This approach lacks a comprehensive understanding of the overall meaning. 2) use data augmentation by generating various chunk-level texts from different structures (Zhang et al., 2022). However, it will produce a large volume of augmented data, adding to the training burden. Given GPT's expertise in processing corpora (Zhou et al., 2024), we design a simple template to help integrate both the title and content into a complete paragraph:



Mutant Ninja Turtles Original Series
User Query
LLM
Last Token Vector From Final Hidden Layer
Training Stage
Soft Prompt
Tunable Prompt Tokens
Transformer Layer
Frozen LLM Parameters
Quantized Prompt-Training
Maximize Aggrement
Inference Stage
Unseen Labels
Fine-tuned LLM
Teenage Mutant Ninja Turtles III, The Manhattan Project
Teenage Mutant Ninja Turtles, Planet of the Turtles
The Lord of the Rings: Return of the King
Harry Potter and the Deathly Hallows
Harry Potter and the Goblet of Fire
Predicted Unseen Labels
Mutant Ninja Turtles Series
Lord Ring Series
Harry Potter Series
Extreme Scale Training Label Set
Hashing Function
Label Subsets
Sampled labels
Diversity-Aware Hash Sampling

Figure 2: The overall framework of our proposed QUEST. First, we select representative query-label pairs from the training set using diverse sampling. These pairs will pass through an encoder in Llama-7b, and the embeddings extracted will be trained through contrastive learning to maximize the similarity between positive pairs. During the inference stage, all labels, including unseen ones, will be processed by a fine-tuned LLM to extract representations, followed by the nearest neighbor search with the target query to choose the top-K items as recommendations.

> **The title of the product is** Lord of the Rings. **Its content is about** the Two Towers action-adventure game that allows players to take control of the trilogy heroes.

The obtained paragraph is first tokenized to form the input sequence and then passed into the large language model denoted as $T = (t_1, t_2, ..., t_{|n|})$. One conventional approach is to use the *cls token* or the average of the last layers as the sentence representation. In contrast, due to the autoregressive nature of generative models, where each predicted word is generated based on the preceding sequence, we extract the last token vector from the final hidden layer as the embedding for the entire sentence.

## 3.2 Utilizing Massive Labels

**Clustered Query-Label Pairs.** The current EMTC pipeline performs CRL (Oord et al., 2018; Gao et al., 2021) to generate representations for input queries and labels. CRL would like to maximize the embedding similarities between relevant pairs and minimize the similarity of irrelevant pairs. Because of the one-to-many matching property in EMTC, it is impractical to train all relevant pairs. Therefore, it is necessary to use sampling techniques. A straightforward solution would be uniform sampling. However, we observe that the ground truth labels are highly clustered. We chose a subset of the training labels based on certain queries. As described in figure 3, it reveals a notable pattern indicating that rather than being randomly distributed, there are several distinct clusters. Hence, we have to perform diversified sampling that deliberately selects representative individuals from diverse subgroups. However, diversified sampling can be computationally intensive in grouping the labels into clusters, especially when we have million-scale labels. Thus we have to explore an effective and efficient diversified sampling strategy to select representative query-label pairs for CRL.

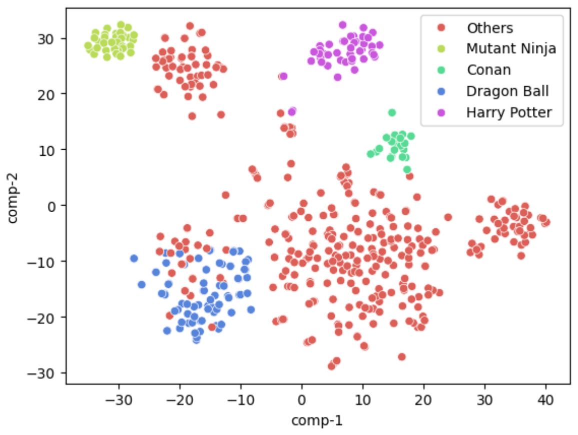


Figure 3: Based on the embeddings produced by SentenceTransformer, the t-SNE visualization of the label distribution indicates that serial books tend to cluster together while others separate apart.

**Diversity-Aware Sampling with Hashing.** We notice a token-level similarity among the various labels for the same query, indicating that label descriptions can be closely clustered. On the other side, for some of the EMTC datasets, we observe that the average number of labels per query is also significant. Given this intensive and clustered label information, we would like to perform a diversity-aware sampling to reduce the data redundancy in CRL for EMTC. Moreover, to address the computational challenges in diversity-aware sampling, we propose an algorithm based on MinHash (Broder, 1997; Broder et al., 1998), which is a locality-sensitive hashing function for Jaccard similarity. MinHash function $h$ maps a set $X$ into an integer in $[B]$, where $B$ is the range of a hash value. Furthermore, we have $\Pr[h(X) = h(Y)] = \frac{|X \cap Y|}{|X \cup Y|}$ for set $X$ and $Y$. Given a dataset of query-label pairs, our algorithm ( see Algorithm 1) first concatenates each query-label pair $(x, y) \in S$ into a single string $s$. Next, we apply $R$ independent MinHash functions to generate $R$ hash values for every $s$. Next, we initialize a zero array with size $R \times B$. For every $s$ and every MinHash function $h_j$, we increment the $j$th row and $h_j(s)$th column with 1. Next, we score each $s$ with its corresponding counts in the array. Finally, we sample $k$ elements from $S$ with the probability defined on the inverse of counts. In practice, $R$ and $B$ have values of 4 and $\log n$ respectively, where $n$ represents the size of training pairs following the theoretical support from (Datar et al., 2004). Intuitively, LSH functions are more likely to map similar text inputs into the same bucket. Thus, buckets with higher counts bring together clustered pairs. Inversely, increasing the probability of searching within minority groups by taking the inverse of counts achieves diversity sampling. As shown in below, Algorithm 1 has a linear time complexity to the size of $S$, outperforming clustering-based approaches with exponential time complexity over the size of $S$. Moreover, the memory complexity is constant. As a result, QUEST provides an efficient algorithm to generate a representative subset of query-label pairs for training.

**Theoretical Justifiaction.** LSH is a randomized function family (Indyk and Motwani, 1998; Datar et al., 2004; Andoni et al., 2014; Andoni and Razenshteyn, 2015; Andoni et al., 2017). Each function in this family maps one input vector into a hash value, usually a binary code or an integer. When vectors are similar, they will likely have the same hash value. The LSH function is defined as:

**Definition 3.1** (Locality-sensitive Hash Family)**.** *We define a function family $\mathcal{H}$ to be $(D, cD, p_1, p_2)$-sensitive with respect to distance function $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ if for any two vector $x, y \in \mathbb{R}^d$, any $h \in \mathcal{H}$ chosen uniformly at random satisfies:*

$$d(x, y) \le D \;\to Pr[h(x) = h(y)] \ge p_1, \quad (1)$$
$$d(x, y) \ge cD \;\to Pr[h(x) = h(y)] \le p_2. \quad (2)$$

The occurrence referred to as a "collision" happens when both $x$ and $y$ end up with an identical hash value. For example, in the case of a series of books, there might be slight variations between versions, but they still maintain similar titles and content, leading to a reduced Euclidean distance. The likelihood of a collision between $x$ and $y$ steadily diminishes following a distance function $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. This is formally depicted as:

$$Pr[h(x) = h(y)] \propto f(d(x, y)). \quad (3)$$

In this work, we would like to design a sketch of training pairs set $S$ that can represent the diversity of each query-label pair $s \in S$. Here we define this "diversity" as the density of the Jaccard kernel. For every query-label pair $q \in S$, we define its Jaccard Kernel Density as follows: $\mathcal{J}(q)$ as

$$\mathcal{J}(q) = \sum_{s \in S} \frac{|\phi(q) \cap \phi(s)|}{|\phi(q) \cup \phi(s)|}. \quad (4)$$

where $\phi$ defines the function that tokenized the query-label pair in text into a set of tokens, however, it is infeasible to compute the precise Jaccard kernel density following the expressions above since it requires $O(|S|^2 \mathbf{NNZ}(S))$ time to compute. Here $|S|$ is the size of set $S$ and $\mathbf{NNZ}(S)$ is the maximum number of elements in set $\phi(s)$ for any $s \in S$. To address this challenge, we efficiently estimate the Jaccard kernel density through the fact that $\mathbb{E}[Sum_q] = R \cdot \mathcal{J}(q)$. The full derivation of this formula can be checked in the appendix A.2.

## 3.3 Quantized-Prompt Training

Current commodity hardware cannot support the training and fine-tuning requirements for LLMs. The memory cost of storing a LLaMA-7B model's weight in FP16 format is already about 14G, taking over half of the memory capacity of an RTX 3090 GPU. As a result, we have limited resources to perform CRL for EMTC on commodity hardware. A variety of memory-efficient fine-tuning (MEFT) methods have emerged for LLMs' adaptation to different tasks on commodity hardware. MEFT strives to fine-tune only a small subset of the model's parameters, achieving desired performance while significantly reducing memory requirements.

QUEST performs a quantized-prompt training with CRL objective defined in Eq (5). In this approach, the pre-trained LLM parameters are quantized into lower bits. Next, QUEST inserts soft prompt tokens before the input query tokens to

**Algorithm 1** Diversity-Aware Sampling

```
Input: Training pairs set $S$ with size $N$, Sample size $k$, $R$ independent MinHash functions $\{h_1, h_2, \cdots, h_R\}$, each with range $B$
Output: $k$ elements from $S$.
Initialize: $C \leftarrow 0^{R\times B}$
Initialize: $p \leftarrow \emptyset$
for query-label pair $s \in S$ do
    for $r = 1 \to R$ do
        $C[r, h_r(s)] += 1$
    end for
end for
for query-label pair $s \in S$ do
    $Sum_s \leftarrow 0$
    for $r = 1 \to R$ do
        $Sum_s \leftarrow Sum_s + C[r, h_r(s)]$
    end for
    $p_s \leftarrow N * R / Sum_s$
    $p \leftarrow p \cup \{p_s\}$
end for
Sample $k$ pairs $S_k$ from $S$, each $s \in S$ has the sampling probability $p_s$.
return $S_k$
```

control the embedding produced by LLM. During the training, only the soft prompt tokens adaptively update while all other quantized parameters in the network remain unchanged. We argue that quantized-prompt training is more memory-efficient so that QUEST can perform CRL for EMTC on commodity hardware such as RTX-3090. We also conduct a quality comparison of QUEST with Q-LoRA (Dettmers et al., 2023) to suggest the difficulty in tuning adapters for EMTC.

Given the sampled pairs generated from the last section, we match the representations of queries and labels and then maximize the cosine similarity between corresponding pairs. Let $H(\cdot)$, chosen as a LLaMA-7B model, represent the embedding function of the text, with $h(x_i)$ and $h(y_i)$ as the embedding vector of the query element and label set element, respectively. As shown in Eq.(5), the objective of the model is to maximize the cosine similarity between $h(x_i)$ and $h(y_i)$.

$$\ell = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{\exp^{sim(h(x_i),h(y_i))}}{\sum_{j=1}^{N} \exp^{sim(h(x_i),h(y_j))}}. \quad (5)$$

After completing the training, we implement the nearest neighbor search. We follow the same process as in the text representation section, concatenating the title and content of the query using a template. Then we input them into the trained model and retrieve the k-nearest embeddings in the set $H(Y)$ as the recommended items. The search process is carried out by the FAISS engine.

## 4 Experiments

We conduct extensive experiments on four real-life datasets. Our study aims to address the following research questions: **Q1:** Can QUEST achieve superior prediction performance than current state-of-the-art frameworks? **Q2:** How effective is hash sampling in reducing dataset size without sacrificing performance and consuming excessive time, compared to random sampling and other clustering techniques? **Q3:** How quickly can QUEST address the EMTC issue in comparison to existing baselines, and to what extent can QUEST be compressed by utilizing various quantization methods?

### 4.1 Experimental Settings

**Datasets.** We evaluate QUEST's performance using the following four datasets: Amazon-131K, Amazon-1M, Wikipedia-500K, and WikiAlsoSee-320K. These four are for product recommendations, scientific terms, and article categorization. All datasets are derived from real-life scenarios, and the details are presented in table 2.

Table 2: Statistics of the datasets. The table shows the number of training data, testing data, and labels.

| Dataset | $X_{train}$ | $X_{test}$ | $Y$ |
|---|---|---|---|
| Amazon-131K | 294,805 | 134,835 | 131,073 |
| Amazon-1M | 914,179 | 1,465,767 | 960,106 |
| Wikipedia | 1,813,391 | 783,743 | 501,070 |
| WikiSeeAlso | 693,082 | 177,515 | 312,330 |

**Evaluation Metrics.** To evaluate the recommendation performance of the framework, we adopt precision and recall rate as the main indicators, which are defined as:

$$\begin{aligned} P@p &= \frac{\sum_{i=1}^{n}\sum_{y\in Y_i^{pred}} \mathbb{1}_i^{top\text{-}p}(y)}{np}, \\ R@r &= \frac{1}{n}\sum_{i=1}^{n} \frac{\sum_{y\in Y_i^{pred}} \mathbb{1}_i^{top\text{-}r}(y)}{\sum_{y\in Y} \mathbb{1}_i^{top\text{-}r}(y)}, \end{aligned} \quad (6)$$

where $n$ represents the number of evaluated items, $Y_i^{pred}$ denotes the target query's label set,

Table 1: Performance comparison on four public datasets.

| Dataset | Metric | GloVe | SimCSE | TF-IDF | MPNet | ICT | MACLR | RTS | QUEST |
|---|---|---|---|---|---|---|---|---|---|
| LF-Amazon-131K | P@1 | 3.67 | 10.13 | 12.38 | 13.94 | 13.82 | 18.13 | 18.74 | **25.48** |
| | P@3 | 2.78 | 8.61 | 11.50 | 11.41 | 11.41 | 15.42 | 15.30 | **17.69** |
| | P@5 | 2.15 | 6.69 | 9.14 | 8.82 | 8.90 | 11.93 | 11.96 | **13.40** |
| | R@1 | 2.05 | 5.61 | 6.91 | 7.82 | 7.76 | 10.35 | 10.64 | **14.61** |
| | R@3 | 4.33 | 13.39 | 18.14 | 18.08 | 18.09 | 24.45 | 24.16 | **27.73** |
| | R@5 | 5.44 | 16.84 | 23.21 | 22.58 | 22.80 | 30.43 | 30.45 | **33.99** |
| | R@10 | 7.23 | 21.27 | 29.32 | 27.91 | 28.94 | 37.28 | 38.19 | **41.96** |
| | R@100 | 14.17 | 35.81 | 45.04 | 43.39 | 47.40 | 54.99 | 59.34 | **64.30** |
| LF-Wikipedia-500K | P@1 | 2.19 | 14.32 | 20.30 | 22.46 | 17.74 | 28.44 | 30.67 | **32.94** |
| | P@3 | 1.52 | 6.84 | 12.98 | 12.87 | 9.67 | 17.75 | 19.03 | **20.44** |
| | P@5 | 1.23 | 4.55 | 9.96 | 9.49 | 7.06 | 13.53 | 14.34 | **15.60** |
| | R@1 | 0.85 | 4.24 | 7.25 | 8.74 | 7.35 | 10.40 | 10.58 | **12.05** |
| | R@3 | 1.66 | 8.03 | 12.91 | 14.07 | 11.60 | 18.16 | 18.48 | **20.44** |
| | R@5 | 2.18 | 11.26 | 15.98 | 16.76 | 13.84 | 22.38 | 22.51 | **25.52** |
| | R@10 | 3.10 | 14.35 | 20.31 | 20.64 | 17.19 | 28.52 | 28.23 | **32.36** |
| | R@100 | 8.52 | 27.68 | 38.16 | 34.72 | 31.08 | 50.09 | 48.00 | **57.07** |
| LF-WikiSeeAlso-320K | P@1 | 3.86 | 9.03 | 10.71 | 13.75 | 10.76 | 16.31 | 18.64 | **26.69** |
| | P@3 | 2.76 | 6.64 | 8.90 | 11.93 | 10.05 | 13.53 | 15.14 | **18.65** |
| | P@5 | 2.21 | 5.22 | 7.15 | 9.58 | 8.12 | 10.78 | 12.07 | **14.71** |
| | R@1 | 2.12 | 4.99 | 5.92 | 8.14 | 6.12 | 9.71 | 10.86 | **14.16** |
| | R@3 | 4.11 | 9.89 | 13.03 | 17.77 | 14.32 | 20.39 | 22.68 | **26.44** |
| | R@5 | 5.22 | 12.34 | 16.48 | 22.21 | 18.05 | 25.37 | 28.29 | **33.02** |
| | R@10 | 6.95 | 15.93 | 21.60 | 28.11 | 23.01 | 32.05 | 35.47 | **42.28** |
| | R@100 | 15.33 | 30.11 | 42.55 | 45.91 | 39.77 | 53.83 | 57.30 | **71.05** |
| LF-Amazon-1M | P@1 | 4.05 | 3.33 | 7.68 | 8.29 | 8.66 | 9.58 | 10.00 | **18.49** |
| | P@3 | 4.07 | 3.69 | 9.20 | 8.87 | 9.26 | 10.41 | 10.95 | **11.71** |
| | P@5 | 3.07 | 2.74 | 7.23 | 6.80 | 7.13 | 8.03 | 8.41 | **8.68** |
| | R@1 | 2.91 | 2.38 | 5.61 | 6.04 | 6.30 | 7.38 | 7.34 | **13.48** |
| | R@3 | 8.42 | 7.66 | 19.30 | 18.64 | 19.45 | 22.01 | 23.09 | **24.49** |
| | R@5 | 10.44 | 9.38 | 24.92 | 23.51 | 24.60 | 27.72 | 29.14 | **29.97** |
| | R@10 | 12.90 | 11.43 | 31.76 | 29.35 | 30.73 | 34.48 | 36.30 | **37.35** |
| | R@100 | 21.18 | 18.54 | 51.79 | 46.15 | 48.42 | 55.23 | 55.84 | **60.59** |

and the indicator function $\mathbb{1}$ counts the instances where the retrieved item belongs to the ground truth recommendations set.

**Implementation Details.** We utilize a server equipped with six 24 GB NVidia RTX 3090 GPUs. Our approach involves employing the Adam optimizer with a learning rate set at 0.001 and implementing early stopping based on the validation set accuracy. The hyperparameter (sampling ratio) $\rho$ is fine-tuned through a grid search, with the optimal hyperparameter selected for each dataset.

### 4.2 Baselines

We utilize several classical techniques for embedding text. These methods encode textual descriptions initially and then retrieve the most similar items from the label set. Here are previews of the baselines that are incorporated:

- **TF-IDF** (Ramos et al., 2003) selects words based on their frequencies in the text.
- **GloVe** (Pennington et al., 2014) makes dense vector representations of words instead of one-hot encoders.
- **SimCSE** (Gao et al., 2021) is a contrastive learning framework for generating sentence embeddings. It takes an input sentence and predicts itself in a contrastive objective, with only standard dropout used as noise.
- **MPNet** (Song et al., 2020) pre-trains language models by combining masked language modeling (MLM) and permuted language modeling (PLM) into a single approach.
- **Inverse Cloze Task (ICT)** (Lee et al., 2019) retrieves evidence candidates to answer open-domain questions using a Bert-based model.

- **MACLR** (Xiong et al., 2021) leverages the raw text using techniques such as Multi-scale Adaptive Clustering, Label Regularization, and self-training with pseudo-positive pairs.

- **RTS** (Zhang et al., 2022) is a framework used for handling titles and content through conducting augmentation from chunk-level texts.

### 4.3 Main Results

The comparison of prediction performance on four datasets between QUEST and other baseline methods is shown in Table 1. To assess the effectiveness of QUEST, we conduct extensive experiments using various text-embedding approaches and the latest methods. The best result for each baseline group is highlighted by underlining.

- QUEST exceeds all benchmarks in precision and recall rates across all datasets. This success is attributed to the utilization of state-of-the-art large language model frameworks. Prompt-tuning and diverse sampling collectively help alleviate the training workload, thereby enabling the training of such a massive framework on a single device.

- Among all the baseline methods, GloVe embeddings fail to produce satisfactory results, while Bag-of-words performs better due to its large dimension and sparse characteristics. Fine-tuned Bert learns a more superior task-specific than pre-trained language models. Our study demonstrates that the remarkable comprehension ability of contemporary generative large language models can be reproduced in the field of embedding learning.

### 4.4 Effectiveness of Diverse Sampling

We compare the average $P@1$, $R@10$, and $R@100$ indicators of hash sampling with random sampling to validate its effectiveness. As depicted in Figure 4, the prediction performance of the training dataset selected by hash sampling surpasses the baseline across different sampling proportions. It is noticeable that the smaller the sampling proportion, the greater the advantage, confirming the significant superiority of our method, especially in scenarios where computational resources are limited. Furthermore, the marginal benefit from increasing the dataset is diminishing, verifying the findings mentioned in the design space section that many training labels are highly similar. Therefore, using all training pairs can only result in little improvement. In practice, the sampling ratio $\rho$ for an extremely large dataset (1M) is selected to be 0.25. **K-means clustering proves to be challenging in this task as it has a quadratic time complexity and it grows significantly as K increases, making it unavailable for processing millions of data points.** In our experiments, we failed to obtain a response using the K-means algorithm. Conversely, our Hash-sampling approach can yield results within minutes.

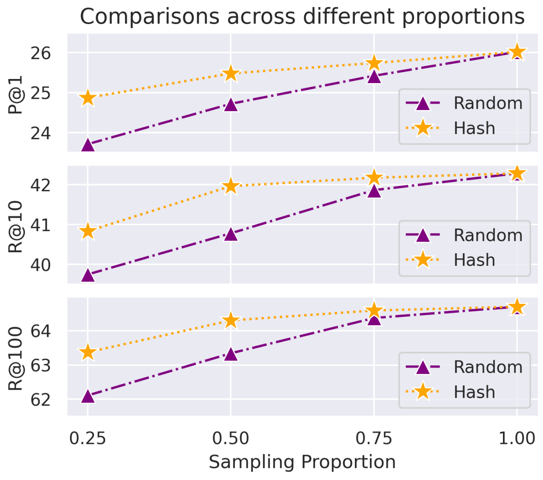


Figure 4: The comparison between hash sampling and random sampling. All y-axis units are in % proportions.

### 4.5 Efficiency Analysis

**Other MEFT methods.** In this section, we want to compare the performance of QUEST and QLoRA with a similar number of parameters (QUEST: 20,480 vs QLoRA: 32,768). We load the trainable adapter towards the last transformer layer. An important finding is that training QLoRA is extremely challenging, especially when the trainable parameters are few. The loss is very difficult to converge (the R@1 is merely 0.20%), indicating that QUEST is much more effective in embedding extraction. Additionally, we tested various quantization techniques. While NF-4 quantization is widely used, there are also alternative variants that employ fewer bits. The outcomes are presented in Table 3. GPTQ compresses trained weights to lower bits by minimizing the mean squared error (Frantar et al., 2022). Alpha-tuning adopts BCQ-format to divide weights into a summation of k elements (Kwon et al., 2022). In a 4-bit scenario, NF-4 performs the best. If users aim for even smaller compression, using 3-bit GPT-Q and alpha-tuning will introduce

Table 3: Comparisons of various quantization methods.

| Quantization | | | |
|---|---|---|---|
| Method | Bits | P@1 | R@100 |
| Bfloat16 | 16-bit | 25.86 | 64.98 |
| NF-4 | 4-bit | 25.48 | 64.30 |
| GPTQ | 4-bit | 25.04 | 62.91 |
| GPTQ | 3-bit | 24.35 | 61.41 |
| GPTQ | 2-bit | 13.83 | 34.01 |
| AlphaTuning | 3-bit | 23.34 | 61.60 |
| AlphaTuning | 2-bit | 10.47 | 25.35 |

some loss, but going down to 2-bit will significantly decrease performance. For a balance between computational efficiency and accuracy, we are using NF-4 compression. In the future, we will consider adopting more advanced compression techniques.

**Time consumption.** All experiments of QUEST were conducted on a single Nvidia 3090 GPU with 24GB memory capacity. Our strongest baseline, RTS, uses 4 cloud V-100 32G GPUs and trains for 30 hours to get the results on the Amazon1M dataset. Our approach requires 48 hours of training. RTS takes higher costs with smaller models due to the need for expensive augmentation over title and content pairs, but we utilize templates to better encode different text structures.

### 4.6 Hyperparameter Study

In our research, the complexity of our framework mainly depends on two aspects: the number of trainable tokens $K$ in the prompt and the length of the tokenized sequence $L$ for each piece of text. The ablation study assesses the recall rate under different combinations of these two variants.

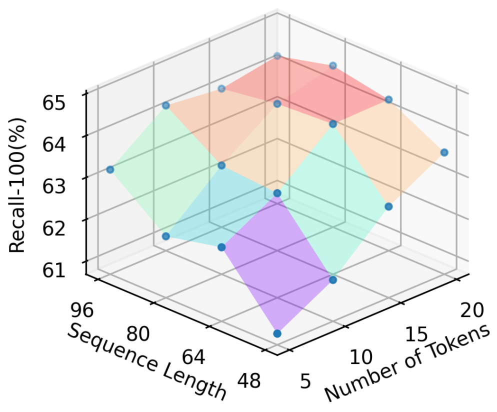


Figure 5: Grid-search results on Amazon-131K.

Figure 5 shows that using a smaller number of prompt tokens and shorter sequence lengths may result in relatively lower performance. However, adding too many tokens or increasing sequence length does not lead to better performance either. The best result was achieved with a configuration of $k$ at 15 and $L$ set to 64. Overall, there is no significant drop in performance, indicating that our model is not excessively hyperparameter-sensitive.

## 5 Related Work

**EMTC tasks** are prevalent in recommendation systems, content tagging, and any application where the set of potential labels is dynamic or extensive. It can handle cold-start problems in e-commerce systems (Li et al., 2019; Chang et al., 2021; Zhou et al., 2020). Existing attempts include using sparse embedding, glove vectors, and transformer-based variants (Bhatia et al., 2015; Pennington et al., 2014; Lin et al., 2017; Gao et al., 2021). On the other hand, efforts have been made from the perspective of the property of labels. Khandagale et al., 2020 learn the tree structure to generalize and partition the representations of labels. GROOV (Simig et al., 2022) employs the sequence-to-sequence model to map inputs to a collection of textual labels. Wei and Li, 2019 argue that infrequently occurring tail labels are less significant than commonly seen labels. Chalkidis et al., 2020 introduce label hierarchy to tackle skewed label distribution. Others leverage text information by clustering and label regularization(Xiong et al., 2021). Word-level augmentation has also proven to be effective in enhancing text embeddings (Luo et al., 2021).

**Parameter-efficient training methods** are techniques that aim to train models with a limited number of parameters while maintaining or enhancing performance. There are several classical algorithms: Low-Rank Approximations reduce the training load by approximating weight matrices with lower ranks(Hu et al., 2021; Xu et al., 2023; Liu et al., 2023; Shengyuan et al., 2024). Knowledge Distillation trains a smaller "student" model to imitate the predictions of a larger "teacher" model (Cho and Hariharan, 2019; Sanh et al., 2019). Quantization decreases the bit-width of model parameters and has been incorporated in recent studies to further accelerate training and compress the model size (Polino et al., 2018; Lin et al., 2023). QLoRA employs 4-bit quantization and lower-rank decomposition to achieve a similar effect as the original model (Dettmers et al., 2023).

## 6 Conclusions

In this paper, we validated the effectiveness of large generative language models in EMTC compared to traditional methods like bag-of-words vectors and Bert-based models. Given the fact that many users cannot afford expensive computational resources, all of our study revolves around this motivation. We employ prompt-tuning, quantization, and hash sampling to mitigate the side effects of the extremely large model size and vast training pairs, enabling training on a single machine within a limited time. The QUEST model manages to handle large classification tasks and comprehensive experiments confirm the efficacy of each module.

## Limitations

In this paper, we observed a significant performance drop with 2-bit quantization. With new quantization techniques emerging, exploring smaller models is worthwhile, and we plan to continue our research in this area. Although GPTQ and Alpha tuning perform well with 3-bit quantization, we could only compress the model post-training. Directly training a 3-bit quantized model proved difficult, as convergence was hard to achieve. There is a significant difference between compressing after training and training with compression. The latter approach allows individuals to train personalized recommendation models on their own devices, which greatly enhances privacy protection.

## Ethics Statement

We all comply with the ACL Ethics Policy[1] in this study. We used open-resource datasets that have been extensively used in prior research. Consumer information has been anonymized and privacy has been carefully protected.

## References

Alexandr Andoni, Piotr Indyk, Huy L Nguyen, and Ilya Razenshteyn. 2014. Beyond locality-sensitive hashing. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1018–1028. SIAM.

Alexandr Andoni, Thijs Laarhoven, Ilya Razenshteyn, and Erik Waingarten. 2017. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 47–66. SIAM.

Alexandr Andoni and Ilya Razenshteyn. 2015. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing (STOC)*, pages 793–801.

K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. 2016. The extreme classification repository: Multi-label datasets and code.

Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. *Advances in neural information processing systems*, 28.

Andrei Z. Broder. 1997. On the resemblance and containment of documents. In *the Compression and Complexity of Sequences*.

Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. 1998. Min-wise independent permutations. In *STOC*.

Ilias Chalkidis, Manos Fergadiotis, Sotiris Kotitsas, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. An empirical study on large-scale multi-label text classification including few and zero-shot labels. *arXiv preprint arXiv:2010.01653*.

Wei-Cheng Chang, Daniel Jiang, Hsiang-Fu Yu, Choon Hui Teo, Jiong Zhang, Kai Zhong, Kedarnath Kolluri, Qie Hu, Nikhil Shandilya, Vyacheslav Ievgrafov, et al. 2021. Extreme multi-label learning for semantic matching in product search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2643–2651.

Shengyuan Chen, Qinggang Zhang, Junnan Dong, Wen Hua, Qing Li, and Xiao Huang. 2024. Entity alignment with noisy annotations from large language models. *arXiv preprint arXiv:2405.16806*.

Jang Hyun Cho and Bharath Hariharan. 2019. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4794–4802.

Benjamin Coleman, Anshumali Shrivastava, and Richard G Baraniuk. 2019. Race: Sub-linear memory sketches for approximate near-neighbor search on streaming data. *arXiv preprint arXiv:1902.06687*.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

[1] `https://www.aclweb.org/portal/content/acl-code-ethics`

Hao Ding, Yifei Ma, Anoop Deoras, Yuyang Wang, and Hao Wang. 2021. Zero-shot recommender systems. *arXiv preprint arXiv:2105.08318*.

Junnan Dong, Qinggang Zhang, Chuang Zhou, Hao Chen, Daochen Zha, and Xiao Huang. 2024a. Cost-efficient knowledge-based question answering with large language models. *arXiv preprint arXiv:2405.17337*.

Junnan Dong, Qinggang Zhang, Huachi Zhou, Daochen Zha, Pai Zheng, and Xiao Huang. 2024b. Modality-aware integration with large language models for knowledge-based visual question answering. *arXiv preprint arXiv:2402.12728*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 604–613, Dallas, TX.

Sujay Khandagale, Han Xiao, and Rohit Babbar. 2020. Bonsai: diverse and shallow trees for extreme multi-label classification. *Machine Learning*, 109:2099–2119.

Se Jung Kwon, Jeonghoon Kim, Jeongin Bae, Kang Min Yoo, Jin-Hwa Kim, Baeseong Park, Byeongwook Kim, Jung-Woo Ha, Nako Sung, and Dongsoo Lee. 2022. Alphatuning: Quantization-aware parameter-efficient adaptation of large-scale pre-trained language models. *arXiv preprint arXiv:2210.03858*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300*.

Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang. 2019. From zero-shot learning to cold-start recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4189–4196.

Yuhan Li, Zhixun Li, Peisong Wang, Jia Li, Xiangguo Sun, Hong Cheng, and Jeffrey Xu Yu. 2023. A survey of graph meets large language model: Progress and future directions. *arXiv preprint arXiv:2311.12399*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. 2017. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 115–124.

Zirui Liu, Chen Shengyuan, Kaixiong Zhou, Daochen Zha, Xiao Huang, and Xia Hu. 2023. Rsc: accelerate graph neural networks training via randomized sparse computations. In *International Conference on Machine Learning*, pages 21951–21968. PMLR.

Dongsheng Luo, Wei Cheng, Jingchao Ni, Wenchao Yu, Xuchao Zhang, Bo Zong, Yanchi Liu, Zhengzhang Chen, Dongjin Song, Haifeng Chen, et al. 2021. Unsupervised document embedding via contrastive augmentation. *arXiv preprint arXiv:2103.14542*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Chen Shengyuan, Yunfeng Cai, Huang Fang, Xiao Huang, and Mingming Sun. 2024. Differentiable neuro-symbolic reasoning on large-scale knowledge graphs. *Advances in Neural Information Processing Systems*, 36.

Daniel Simig, Fabio Petroni, Pouya Yanki, Kashyap Popat, Christina Du, Sebastian Riedel, and Majid Yazdani. 2022. Open vocabulary extreme classification using generative models. *arXiv preprint arXiv:2205.05812*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867.

Thomas N Theis and H-S Philip Wong. 2017. The end of moore's law: A new beginning for information technology. *Computing in science & engineering*, 19(2):41–50.

Tong Wei and Yu-Feng Li. 2019. Does tail label help for large-scale multi-label learning? *IEEE transactions on neural networks and learning systems*, 31(7):2315–2324.

Yuanhao Xiong, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, and Inderjit Dhillon. 2021. Extreme zero-shot learning for extreme text classification. *arXiv preprint arXiv:2112.08652*.

Zhaozhuo Xu, Zirui Liu, Beidi Chen, Yuxin Tang, Jue Wang, Kaixiong Zhou, Xia Hu, and Anshumali Shrivastava. 2023. Compress, then prompt: Improving accuracy-efficiency trade-off of llm inference with transferable prompt. *arXiv preprint arXiv:2305.11186*.

Tianyi Zhang, Zhaozhuo Xu, Tharun Medini, and Anshumali Shrivastava. 2022. Structural contrastive representation learning for zero-shot multi-label text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4937–4947.

Huachi Zhou, Hao Chen, Junnan Dong, Daochen Zha, Chuang Zhou, and Xiao Huang. 2023. Adaptive popularity debiasing aggregator for graph collaborative filtering. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 7–17.

Huachi Zhou, Shuang Zhou, Hao Chen, Ninghao Liu, Fan Yang, and Xiao Huang. 2024. Enhancing explainable rating prediction through annotated macro concepts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11736–11748.

Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. 2020. Towards deeper graph neural networks with differentiable group normalization. *Advances in neural information processing systems*, 33:4917–4928.

## A Appendix

### A.1 Quantization

**Alpha-tuning** adopts BCQ format to approximate $w$ to be summation of $q$ multiplications between $\alpha$ and $b$ where $q$ is the number of quantization bits, $\alpha$ is a scaling factor and $b \in \{-1, +1\}$ is a binary vector.

$$W \approx \sum_{i=1}^{q} \text{diag}(\alpha_i) \cdot B_i.$$

For one-bit quantization, the analytic solution to minimizing $\|w - \alpha b\|^2$:

$$b^* = \text{sign}(w),$$

$$\alpha^* = \frac{w^\top b^*}{n}.$$

For multiple bits quantization, we apply the greedy approximation method to first produce $\alpha_1$ and $b_1$ from the equation above and then iteratively update $\alpha_i$ and $b_i$ through the following functions:

$$b_i^* = \text{sign}(w - \sum_{j=1}^{i-1} \alpha_j b_j),$$

$$\alpha_i^* = \frac{w - \sum_{j=1}^{i-1} \alpha_j b_j^\top b_i^*}{n}.$$

**NF4** is a 4-bit data type used in machine learning, normalizing each weight to a value between -1 and 1 to provide a precise representation of lower precision weight values in contrast to a standard 4-bit float. It utilizes a two-step quantization to enhance memory efficiency for each linear layer.

$$Y^{BF16} = X^{BF16} doubleDequant(c_1^{FP32}, c_2^{k-bit}, W^{NF4}) + X^{BF16} L_1^{BF16} L_2^{BF16},$$

where doubleDequant($\cdot$) is defined as:

$$doubleDequant(c_1^{FP32}, c_2^{k-bit}, W^{k-bit}) = dequant(dequant(c_1^{FP32}, c_2^{k-bit}), W^{4bit}) = W^{BF16}$$

**GPTQ** is a post-training quantization method. It is built based on the classic OBQ algorithm. In OBS, the authors hope to find a way to erase a weight denoted as $q$, such that the overall error increases minimally, while simultaneously calculating a compensation $\delta_q$ to apply to the remaining weights so that the increased error from erasing this weight is offset. The authors have found such a method, with the formula:

$$w_q = argmin_q \frac{w_q^2}{[H_{qq}^{-1}]},$$

$$\sigma_q = -\frac{w_q}{[H_{qq}^{-1}]} \cdot H_{:,q}^{-1}.$$

Instead of setting timing parameters to zero, GPTQ aims to round weight $w_q$ to the nearest value on the quantization grid where $F$ denotes the set of remaining full-precision weights and $\sigma_F$ is the corresponding optimal update.

$$w_q = argmin_q \frac{(quant(w_q) - w_q)^2}{[H_F^{-1}]_{qq}},$$

$$\sigma_F = -\frac{w_q - quant(w_q)}{[H_F^{-1}]_{qq}} \cdot (H_F^{-1})_{:,q}.$$

### A.2 Hash Sampling

We show that $\mathbb{E}[Sum_q] = R \cdot \mathcal{J}(q)$.

$$\begin{aligned} \mathbb{E}[Sum_q] &= R \cdot \sum_{s \in S} \Pr_{h \sim \mathcal{H}} [h(q) = h(s)] \\ &= R \cdot \sum_{s \in S} \frac{|\phi(q) \cap \phi(s)|}{|\phi(q) \cup \phi(s)|} \\ &= R \cdot \mathcal{J}(q) \end{aligned}$$

where the first step follows from Theorem 2 in (Coleman et al., 2019), the second step follows the definition of MinHash (Broder, 1997). As a result, we can show that Algorithm 1 is essentially using an estimated Jaccard kernel density as a measure of diversity.