Dense Retrieval with Quantity Comparison Intent

Prayas Agrawal
IIT Bombay

Nandeesh Kumar K M
Flipkart

Muthusamy Chelliah
Flipkart

Surender Kumar
Flipkart

Soumen Chakrabarti
IIT Bombay

Abstract

Pre-trained language models (PLMs) fragment numerals and units that express quantities in arbitrary ways, depending on their subword vocabulary. Consequently, they are unable to contextualize the fragment embeddings well enough to be proficient with dense retrieval in domains like e-commerce and finance. Arithmetic inequality constraints ("laptop under 2 lb") offer additional challenges. In response, we propose DeepQuant, a dense retrieval system built around a dense multi-vector index, but carefully engineered to elicit and exploit quantities and associated comparison intents. A novel component of our relevance score compares two quantities with compatible units, conditioned on a proposed comparison operator. The uncertain extractions of numerals, units and comparators are marginalized over in a suitable manner. On two public and one proprietary e-commerce benchmarks, DeepQuant is both faster and more accurate than popular PLMs. It also beats several competitive sparse and dense retrieval systems that do not take special cognizance of quantities.

1 Introduction

Transformer-based contextual text encoders (Devlin et al., 2019; Lewis et al., 2019; Reimers and Gurevych, 2019) have greatly enhanced dense retrieval (Mitra and Craswell, 2018; Zhao et al., 2024) in recent years. In several retrieval applications, such as in finance and e-commerce, *quantities* enjoy a central role. A quantity consists of a *numeric* component and possibly a *unit*, which implies the type of the quantity. E.g., a laptop may weigh `2.5` (numeral) `lbs` (unit of weight). Units can be written in many ways (pound, lb, ft, feet). Numerals can have diverse styles (use of comma vs period, or mantissa-exponent vs decimal).

Current transformer-based text encoders use byte-pair or subword (Gage, 1994; Sennrich et al., 2016) dictionaries to fragment the input text. Because this process is not tailored to quantities, numerals and possibly units are fragmented, and transformer networks have limited means to contextualize the fragments back into meaningful representations, resulting in limited *numeracy* (number representations that are effective for inference and generation) (Wallace et al., 2019).

Additional complications arise in quantity cognizant search, because the quantities in the query are often associated with *comparison intents*. These are exceedingly common in e-commerce, e.g., `XL shirt under 20 dollars`, `2-way 6 amp switch`, or `phone with 10 hour battery`. In the first query, the inequality constraint is explicit; in the third, a tablet with a 12-hour battery will be satisfactory (if not too expensive). In the second query, `2` requires a categorical match. Such "common sense" is often baked into PLMs. A PLM may even be able to model that a light car weighs about a ton, whereas a light laptop weighs under two pounds. If only PLMs were endowed with better quantity representations, *and these could be exploited in search*, such queries would perform better.

Our goal is to build a dense retriever with modest-sized PLM as its encoder to fetch documents matching the quantitative inequality constraints in the natural language queries. We wish to capture the semantics of quantities in the same dense space as ordinary words, on both the query and corpus sides.

Our contributions: We present DeepQuant, a quantity and comparison-cognizant dense retrieval system. Like several recent quantity-oriented retrievers, DeepQuant preprocesses the quantity spans in a custom module before injecting quantity-derived features into a transformer-based encoder PLM. The PLM is fine-tuned to reconstruct masked quantities and their units, as well as contextual arithmetic comparison opera-

Findings of the Association for Computational Linguistics: ACL 2025, pages 23825–23839

tors expressed in natural language. Through an attention-like structure that assists diagnostics and explainability, quantity constraints in the query are matched and scored against quantities in corpus documents. DeepQuant uses generic neural building blocks that are end-to-end trainable from relevance judgments. Apart from two public data sets, we experiment with a proprietary data set (ECom) from a major e-commerce company, showing broad gains against sparse and dense retrievers and fine-tuned LMs without special devices to handle quantities, and a large zero-shot PLM. Anonymized prototype code and data are available at `https://anonymous.4open.science/r/proj-D9FE/`

2 Related work

An emerging body of work (Ho et al., 2019, 2020, 2021, 2022; Li et al., 2021; Almasian et al., 2024b) seeks to address the gap between text indexes and quantity-rich queries. Underlying lexical search (sparse retrieval) or execution engines (e.g., Apache Solr, SPARQL implementations) at last stage inference of these systems provide limited *structured* query interfaces (like SQL) involving quantities and hence are not suited for end-user interaction.

Open-domain question answering, where the answer is a distribution over quantities, has been attempted on unstructured text (Roy et al., 2015) and tables with quantities (Ibrahim et al., 2016). Semantic interpretation, e.g., text2sql (Katsogiannis-Meimarakis and Koutrika, 2023), is applicable more for a structured (e.g., RDBMS) rather than unstructured text corpus. In specific verticals such as agronomics, strict schema-adherent data can be extracted from text (Rybiński et al., 2023, SciHarvester) for structured query execution. Also, none of these works addresses quantity comparisons.

Representing numbers meaningfully in language models (LM) is a parallel development (Geva et al., 2020; Sundararaman et al., 2020; Thawani et al., 2021; Liang et al., 2022). Chen et al. (2023, ComNum) prepared a dataset for comparing numbers, which can be used to fine-tune LMs. Numeracy can be enhanced by laying anchors via unsupervised pre-training (Sharma et al., 2024). Alternatively, mathematical priors can help compute aggregated digit embeddings explicitly incorporating them into transformer models (Sivakumar and Moosavi, 2025).

DeepQuant incorporates insights from Spokoyny et al. (2021), who adopted the (mantissa, exponent, unit, type) representation as a 'measurement' and pioneered reconstruction of masked measurements from context. They were not concerned with retrieval. Apropos of retrieval, apart from a disjoint comparison of text and quantity, Almasian et al. (2024a, Section 3.3) propose a templated corpus augmentation method that highlights comparisons and unit surface forms. This is a data engineering approach: once a LM is tuned on this augmented corpus, scoring remains the same as in BM25 (Jones, 1972), ColBERT (Santhanam et al., 2022) or SPLADE (Formal et al., 2021). In contrast, we *engineer both the encoding and scoring networks* substantially, with clear payoffs.

3 DeepQuant

We describe DeepQuant in detail, guided by Figure 1. To keep it simple, most of the exposition will make the assumption that the query has exactly one quantity associated with one comparison intent. This is the case in all but a vanishing minority of queries in all our data sets. However, DeepQuant applies equally to multiple comparisons; see Appendix G.

3.1 Notation and problem statement

The query is denoted Q. Depending on context, we may regard Q as a set, bag (multiset), or sequence of tokens, or a set of contextualized token embeddings. A query token is denoted q, and its contextualized token embedding is denoted $\boldsymbol{q}$. Similarly, a document or passage in the corpus is denoted D, containing tokens d and their embeddings $\boldsymbol{d}$.

While basic sparse and dense retrievers consider all these tokens at par, we will partition each bag (say, $\{\!\{Q\}\!\}$) into two kinds of tokens: ordinary text tokens $\{\!\{Q_\diamond\}\!\}$ and quantity-expressing tokens $\{\!\{Q_\#\}\!\}$. D is subjected to the same procedure. However, the encoder we use will contextualize all tokens together, in their original sequential order.

As in classical learning to rank (Liu, 2009), the retriever's scoring module $\mathrm{score}(D|Q)$ will be trained using a limited corpus of manually labeled relevance judgments. Given a query Q and relevant document $D_\oplus$, we sample some number of

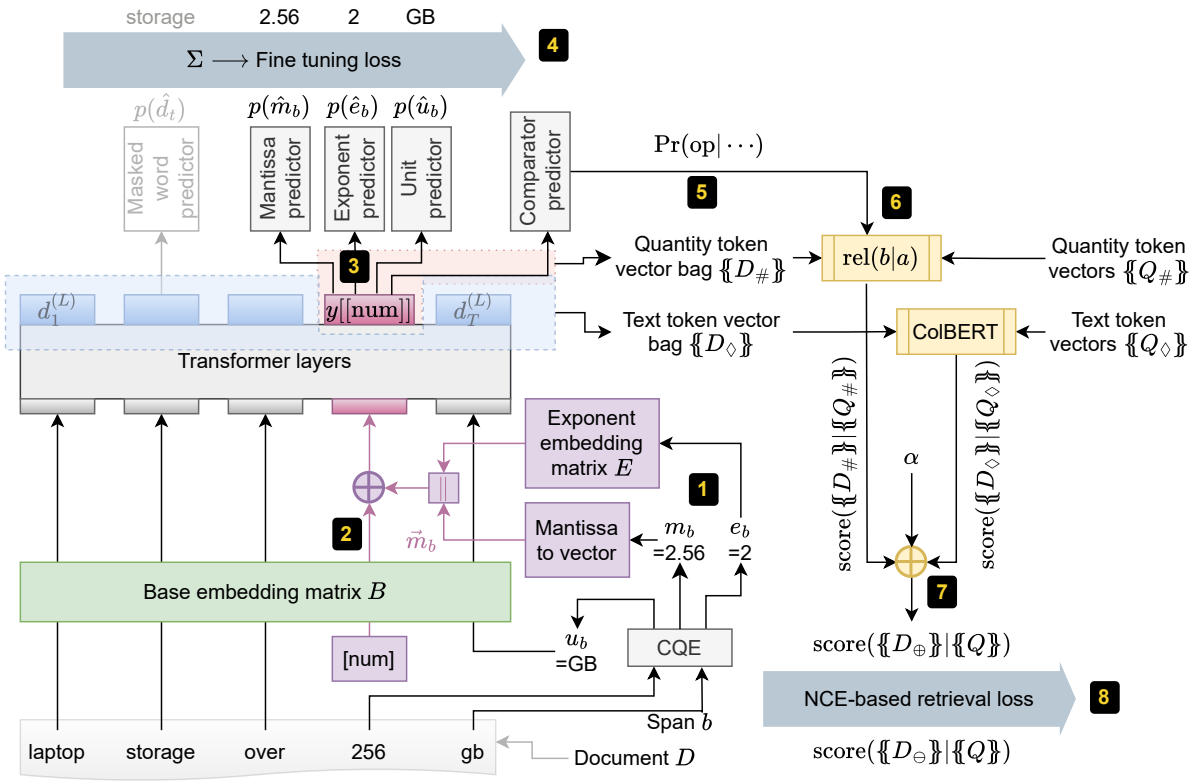

Figure 1: DeepQuant block diagram. (LM tokenization is elided to reduce clutter.) 1 Quantity span b in document D is processed by CQE (or any other accurate quantity extractor) to extract mantissa, exponent and unit. 2 Quantity-specific embeddings are slipstreamed/superposed on a special token `[num]` embedding before being injected into the transformer encoder. 3 Prediction networks for mantissa, exponent, unit and comparison operator are attached to the output embedding for `[num]`. 4 These prediction networks are trained while fine-tuning the PLM. 5 Predicted comparator op is used to define the relevance 6 of span b to analogous quantity span a in the query. 7 Textual and quantity-based relevance scores are combined. 8 The overall score is trained end-to-end using more relevant documents $D_\oplus$ and noise-contrastive samples of less relevant documents $D_\ominus$ for query Q.

other documents $D_\ominus$ presumed less relevant, and use these in a noise-contrastive (Józefowicz et al., 2016) retrieval loss $\mathcal{L}_{\text{retr}}$. During inference, documents will be retrieved in decreasing order of their scores. (Interactive chat systems such as ChatGPT, after this retrieval step, will typically use the top-scoring documents as part of the generation context, but we focus only on the retriever stage.)

3.2 Injecting quantity spans into the encoder

The first step in DeepQuant's encoder pipeline is to extract quantities from queries and documents. We rely on a state-of-the-art quantity extractor, CQE (Almasian et al., 2023), and adapt its output to obtain a (mantissa, exponent, unit) triple. E.g., for input `laptop storage 256gb`, CQE will identify the numeral span `256`, unit span `gb` and extraction triple[1] $(m_s = 2.56, e_s = 2, u_s = \texttt{GB})$.

A standard transformer-based encoder might have tokenized[2] the above text as `lap, top, stor, age, 25, 6, gb`. In DeepQuant, we replace `"256"` with special token `[num]`, which is no longer fragmented by the transformer's tokenizer. In other words, the tokenized input will be `lap, top, stor, age, [num], GB`. Tokens x other than `[num]` are looked up as usual in a base embedding matrix $\boldsymbol{B}$ and vector $\boldsymbol{B}[x, :]$ injected into the transformer.

However, at the `[num]` token, we create a new vector to inject into the transformer by adding together a base embedding for the extracted exponent and mantissa. Exponent e_s is clipped to $\hat{e}_s \in [-20, 20]$, and its corresponding exponent embedding is looked up in another embedding matrix as $\boldsymbol{E}[\hat{e}_s, :]$. For representing the mantissa we choose a prototype based strategy (Sundararaman et al., 2020), with a Gaussian kernel. The real number m_s is thereby mapped to a vector akin to a positional embedding $\vec{m}_s$. Thus, the vector injected into the transformer at token `[num]` will be $\boldsymbol{B}[\texttt{[num]}, :] + (\boldsymbol{E}[\hat{e}_s, :] \| \vec{m}_s)$ where '‖' denotes concatenation.

[1] `256` is canonicalized to 2.56×10^2 and `gb` to `GB`.
[2] Subword splice '`##`' elided to reduce clutter.

3.3 Training loss for quantities

We extend NumGPT (Jin et al., 2021) to define a training loss for quantities, because, as they showed, token-based pretraining does not find good representations for numbers. We reused their loss components for the mantissa and exponent, and add a third component for the unit. The transformer encoder will output a contextualized vector for the input token `[num]`, which we call $\boldsymbol{y}$[`[num]`, :]. Note that this is expected to include unit clues, if any, in the input text. E.g., if the input were `lap, top, stor, age, [num], gb`, the representation of token `"gb"` would be suitably assimilated into the output vector for token `[num]`. This output vector $\boldsymbol{y}$[`[num]`, :] is input to three output networks, which produce logits for the mantissa, exponent and unit. Losses for mantissa and exponent are computed as by Jin et al. (2021); loss for the unit is standard categorical cross-entropy. Collectively, we call these losses $\mathcal{L}_{\text{quant}}$.

3.4 Comparison operator prediction

We denote the query (text) as Q and candidate document/passage (text) as D. Suppose a, b are `[num]` tokens in Q, D respectively. Let $\boldsymbol{y}_a, \boldsymbol{y}_b$ be the contextualized output vectors at these positions. In principle, the relevance of D wrt Q depends on comparison hints in Q and the specific quantities in Q and D.

E.g., for $Q =$ `laptop under $600`, $D_{\oplus} =$ `Lenovo ThinkPad 512GB storage 560 USD` is more relevant than $D_{\ominus} =$ `Apple MacBook 128 GB price $1200`.

To estimate relevance, it is important to associate, say, `$600` with a comparison operator (or comparator), from among op$\in \{<, =, >\}$ — here, the searcher wants "$<$ \$600".

Comparator prediction amounts to estimating $\Pr(\text{op} \mid \boldsymbol{y}_a, \boldsymbol{y}_b)$. In principle, comparators can be specified in both Q and D. In practice, documents, being declarative, will usually state quantities such as `$560` explicitly, rather than assert comparisons over known quantities. Therefore, we simplify comparator prediction to the estimation of $\Pr(\text{op} \mid \boldsymbol{y}_a)$. This decision is guided by our motivating applications, and also reduces the need for early interaction. We model

$$\Pr(\text{op} \mid \boldsymbol{y}_a) = \underset{\{<,=,>\}}{\text{SoftMax}}(N_o(\boldsymbol{y}_a)), \qquad (1)$$

where N_o is a simple feed-forward network that outputs 3-way logits over the label space $\{<, =, >\}$. Our relevance score will be defined as a differentiable function of $\Pr(\text{op} \mid \boldsymbol{y}_a)$, and we will be able to train N_o end-to-end with distant supervision from $\langle Q, D_{\oplus}, D_{\ominus} \rangle$ without needing direct supervision of op.

3.5 Scoring numeral pairs wrt a comparator

As before, suppose a, b are `[num]` tokens in Q, D respectively, and let $\boldsymbol{y}_a, \boldsymbol{y}_b$ be their contextualized representations. Given a proposed operator op, we wish to score the belief in the claim "b op a". E.g., if $Q =$ `laptop under $600`, then a corresponds to `600 USD` and op$=<$, so if b corresponds to `500 USD`, belief in "b op a" should be high. At a high level, we compute this belief using a network $N(\text{op}, \boldsymbol{y}_a, \boldsymbol{y}_b) \in [0, 1]$. More specifically, we have three networks $N_{\text{op}}(\boldsymbol{y}_a, \boldsymbol{y}_b) \in [0, 1]$, each with separate trainable parameters (details in the Appendix). However, the decisions made by these three networks should not be independent. If $N_<(\boldsymbol{y}_a, \boldsymbol{y}_b) \approx N_>(\boldsymbol{y}_a, \boldsymbol{y}_b)$, we expect $N_=(\boldsymbol{y}_a, \boldsymbol{y}_b)$ to be large. This is encouraged by introducing the loss term:

$$\mathcal{L}_{\text{reg}} = |1 - N_=(\boldsymbol{y}_a, \boldsymbol{y}_b)| \times \exp\left(-|N_<(\boldsymbol{y}_a, \boldsymbol{y}_b) - N_>(\boldsymbol{y}_a, \boldsymbol{y}_b)|^2\right). \qquad (2)$$

3.6 Supplementary training for numeral comparisons

As in math word problems (Sundaram et al., 2022), spreadsheet manipulation (Ma et al., 2024) and semantic interpretation (Zhang et al., 2023), quantities in training corpora may evince strong clusters and low numeric diversity (e.g., prices). However, we need to prepare DeepQuant for far more diversity when deployed, particularly in the query stream (e.g., `apartment under $337k` or `schools accepting gre 327`). Since quantity spans, values and comparison outcomes are known during fine-tuning, we impose the following cross-entropy loss $\mathcal{L}_{\text{comp}}$

$$= \sum_{a,b} \sum_{\text{op}} -[\![a \text{ op } b]\!] \log N(\text{op}, y_a, y_b), \qquad (3)$$

where $[\![a \text{ op } b]\!] = 1$ when the comparison holds true, and 0 otherwise. It is important to note that this loss is computed without regard for units, and only numbers. For example for query `Phones 128GB` and document `Samsung $300`, even the pair (128, 300) contributes to the loss above. This approach is similar to providing algorithmic demonstrations (Veličković and Blundell, 2021; Veličković et al., 2022), and the architecture of DeepQuant helps us integrate it seamlessly.

3.7 Scoring compatibility of units

Comparing numbers in Q and D makes sense only if these numbers have compatible units. E.g., the size of laptop memory in Q should not be compared to its wattage in D. Recall that one of the three output networks after the encoder is trained to predict the unit for each `[num]` token position, say a. Call the output of this network $\boldsymbol{u}_a = N_u(\boldsymbol{y}_a)$. We will compare $\boldsymbol{u}_a, \boldsymbol{u}_b$ as follows, to find unit compatibility:

$$\text{rel}(u_b|u_a) = \frac{\exp(\boldsymbol{u}_a \cdot \boldsymbol{u}_b)}{\sum_{b' \in D} \exp(\boldsymbol{u}_a \cdot \boldsymbol{u}_{b'})}. \quad (4)$$

By design, $\sum_{b \in D} \text{rel}(u_b|u_a) = 1$. In other words, units inferred from quantities in the document compete to match a unit inferred from one quantity in the query. This attention-like unit compatibility score will be useful when we define the overall quantity relevance score.

3.8 Quantity relevance score

Based on our motivating applications, we make the following assumption: for a given quantity mention a in the query, inspection of at most one quantity mention b in the document is adequate to compute how well b satisfies any necessary comparisons with a. (More general scenarios are discussed in Section 6.)

The first step is to combine numeral relevance and unit relevance into

$$\text{rel}(u_b|u_a) N_{\text{op}}(\boldsymbol{y}_a, \boldsymbol{y}_b), \quad (5)$$

where the first term is an estimate of unit compatibility and the second term scores the numeric comparison constraint.

Because we do not know the comparator for sure, our second step must marginalize over possible comparators associated with mention a, to get $\text{rel}(b|a) =$

$$\sum_{\text{op} \in \{<,=,>\}} \underbrace{\Pr(\text{op}\,|\boldsymbol{y}_a)}_{\dagger} \underbrace{\text{rel}(u_b|u_a)\, N_{\text{op}}(\boldsymbol{y}_a, \boldsymbol{y}_b)}_{\S} \quad (6)$$

This formulation decomposes the relevance score into two components: † captures the comparator probability conditioned on the context of a, while § captures the comparator-conditioned relevance score of the quantity pair.

3.9 Multi-vector relevance score

We start with and enhance a multi-vector dense retrieval system such as ColBERT (Santhanam et al., 2022), where Q and D is each represented by a bag of contextual token embeddings, which we denote as $\{\!\{Q\}\!\}$ and $\{\!\{D\}\!\}$. Given Q, ColBERT assigns a score to D as

$$\text{score}(D|Q) = \sum_{\boldsymbol{q} \in \{\!\{Q\}\!\}} \max_{\boldsymbol{d} \in \{\!\{D\}\!\}} \text{sim}(\boldsymbol{q}, \boldsymbol{d}), \quad (7)$$

where no distinction is made between quantities and other tokens.

In DeepQuant, we partition these vector bags into two bags, one for numeral tokens, the other for other text tokens:

$$\{\!\{Q\}\!\} = \{\!\{Q_\#\}\!\} \cup \{\!\{Q_\lozenge\}\!\}, \quad (8)$$

$$\{\!\{D\}\!\} = \{\!\{D_\#\}\!\} \cup \{\!\{D_\lozenge\}\!\}. \quad (9)$$

Bag similarity in DeepQuant is defined as a linear combination of ColBERT similarities between numeral bags and text bags:

$$\begin{aligned}\text{score}(\{\!\{D_\#\}\!\} \cup \{\!\{D_\lozenge\}\!\} \,|\, \{\!\{Q_\#\}\!\} \cup \{\!\{Q_\lozenge\}\!\}) \\ = (1-\alpha)\,\text{score}(\{\!\{D_\#\}\!\} \,|\, \{\!\{Q_\#\}\!\}) \\ + \alpha\,\text{score}(\{\!\{D_\lozenge\}\!\} \,|\, \{\!\{Q_\lozenge\}\!\}). \quad (10)\end{aligned}$$

The quantity-based similarity score is defined using Equation (6), as $\text{score}(\{\!\{D_\#\}\!\} \,|\, \{\!\{Q_\#\}\!\})$

$$= \sum_{a \in \{\!\{Q_\#\}\!\}} \max_{b \in \{\!\{D_\#\}\!\}} \text{rel}(b|a). \quad (11)$$

Meanwhile, $\text{score}(\{\!\{D_\lozenge\}\!\} \,|\, \{\!\{Q_\lozenge\}\!\})$ is computed as in ColBERT.

In (10), α is not a global hyperparameter, but a learnt weight parameter that balances the contribution of word and quantity similarities for each query, because different queries require different weightings—some rely more on textual/semantic similarity, while others depend more on numerical relevance. We use

$$\alpha = \sigma(N_\alpha \cdot \boldsymbol{y}(Q)[\texttt{CLS}]), \quad (12)$$

where $\boldsymbol{y}(Q)[\texttt{CLS}]$ is the `[CLS]` vector after encoding Q, and σ is the sigmoid activation.

3.10 Overall fine-tuning loss

During fine-tuning, the whole PLM is updated, along with other trainable parameters in the networks $N_<, N_=, N_>, N_o, N_u, N_\alpha$, as well as embedding matrices $\boldsymbol{B}$ and $\boldsymbol{E}$. The loss used for fine-tuning has several parts: (a) retrieval loss $\mathcal{L}_{\text{retr}}$, (b) quantity reconstruction loss $\mathcal{L}_{\text{quant}}$ (Section 3.3), (c) regularization between quantity comparisons $\mathcal{L}_{\text{reg}}$ (Section 3.5), and (d) supplementary loss for quantity comparisons $\mathcal{L}_{\text{comp}}$ (Section 3.6).

4 Experiments

We report on experiments with two public and one proprietary datasets, guided by the following research questions.

RQ1: How does DeepQuant compare with sparse and dense retrievers that do not implement special treatment of quantities?
RQ2: How does DeepQuant compare with pre-trained LMs in zero-shot mode?
RQ3: How do competitors compare when queries are bucketed by quantity magnitudes, and by α (that signifies the importance of quantities)?

Dataset	Corpus	Train	Dev	Test
FinQuant	300k	200k	10k	420
MedQuant	150k	150k	10k	210
ECom	100k	60k	5k	10k

Table 2: Train/test split of datasets.

4.1 Datasets

We evaluate our method on the recently-introduced public datasets FinQuant and MedQuant (Almasian et al., 2024a). FinQuant is created from a set of news articles in the categories of economics, science, sports and technology, collected between 2018 and 2022. MedQuant contains TREC Medical Records on clinical trials. Additionally, we report results from a proprietary benchmark 'ECom' from a major e-commerce company. These datasets are consistent with our focus on quantity comparison-constrained *retrieval*, unlike complex question answering benchmarks that test knowledge graph traversal, arithmetic, logic, or aggregation.

4.2 Baselines

We compare DeepQuant against a spectrum of baseline retrievers. In what follows, systems QBM25, QColBERT and QSPLADE are from (Almasian et al., 2024a). We use the sparse retriever workhorse BM25 (Jones et al., 2000) and QBM25, its quantity cognizant extension. For natural dense retriever baselines, we use ColBERT (Santhanam et al., 2022) and QColBERT, where text relevance is computed as in ColBERT, but numeric relevance uses a sparse scoring style. SPLADE (Formal et al., 2021) uses contextual embeddings to compute sparse indices, getting the best of both worlds. QSPLADE is an extension of SPLADE that computes quantity-based scores "on the side" and combine with text relevance.

We also compare with off-the-shelf LMs. We use RankZephyr (Pradeep et al., 2023), fine-tuned for retrieval, starting with Mistral7B. RankZephyr was evaluated in itemwise and listwise mode. In itemwise mode, $\langle Q, D \rangle$ are the inputs and RankZephyr generates a relevance score. Items are sorted by decreasing score. In (the much slower) listwise mode, Q and candidate documents are the inputs and RankZephyr directly outputs a reranking of the candidates.

Unlike DeepQuant's LM and RankZephyr, the largest public LLMs like `gpt-4o-mini` are not possible or practical to fine-tune in-house. Owing to extremely slow response and budget constraints, limited experiments were conducted with `gpt-4o-mini` to get an idea of what might be the best possible LLM performance.

4.3 Implementation details

We use BERT (Devlin et al., 2019) in all variations of DeepQuant. Similar to Almasian et al. (2024a), we report precision P@10, mean reciprocal rank MRR@10, NDCG@10 and recall R@100. All models are fine-tuned with a batch size of 128 and Adam optimizer (Kingma and Ba, 2017) using the default parameters provided by HuggingFace Transformers (Wolf et al., 2020). Fine-tuning is carried on for up to 8 epochs, choosing the best checkpoint based on validation instances. We conducted all in-house PLM experiments on NVIDIA RTX A6000 GPUs with 48GB RAM.

4.4 Overall comparison

Table 3 compares DeepQuant with the baselines described above. We observe that DeepQuant exceeds all baselines, both for precision at top-10 ranks as well as high-recall goals (at rank 100). Many baselines do not treat words and numbers differently. QBM25, QSPLADE, QColBERT assume independence between quantities and words ignoring semantic information related to quantity tokens. For a detailed example, see Appendix E. DeepQuant shows 7% absolute MRR improvement beyond the second-best method on MedQuant and 4% gain on FinQuant. In both datasets, documents contain multiple quantities with the same units, which can confuse retrievers that do not jointly contextualize text and quantities.

An LLM experiment with RankZephyr-7B in a zero-shot setting, listwise ranking though better than pointwise still lags behind several other baselines while taking ≈12 hours on a single GPU—substantially higher than the ≈20 minutes needed by other baselines. This indicates even larger

Model	FinQuant				MedQuant				ECom			
	P@10	M@10	N@10	R@100	P@10	M@10	N@10	R@100	P@10	M@10	N@10	R@100
BM25	0.06	0.15	0.1	0.47	0.05	0.11	0.08	0.37	0.02	0.01	0.01	0.03
SPLADE	0.21	0.51	0.42	0.74	0.15	0.37	0.29	0.63	0.12	0.27	0.24	39
ColBERT	0.24	0.57	0.44	0.77	0.18	0.47	0.36	0.71	0.12	0.29	0.26	0.40
QBM25	0.20	0.53	0.41	0.55	0.18	0.44	0.37	0.51	0.02	0.02	0.01	0.02
QSPLADE	0.30	0.68	0.53	0.83	0.19	0.52	0.38	0.70	0.13	0.28	0.26	0.41
QColBERT	0.30	0.69	0.56	0.87	0.18	0.51	0.37	0.73	0.12	0.30	0.25	0.41
RankZephyr-												
Itemwise	0.16	0.39	0.31	0.69	0.09	0.24	0.19	0.57	–	–	–	–
Listwise	0.23	0.55	0.45	0.69	0.15	0.36	0.32	0.60	–	–	–	–
GPT-4o-mini	0.17	0.52	0.36	0.69	0.13	0.36	0.26	0.62	–	–	–	–
DeepQuant	**0.32**	**0.73**	**0.59**	**0.88**	**0.21**	**0.59**	**0.44**	**0.80**	**0.15**	**0.35**	**0.29**	**0.42**
	+6.7%	+5.8%	+5.4%	+1.1%	+10.5%	+12.7%	+15.8%	+9.6%	+13.3%	+16.6%	+11.5%	+2.4%

Table 3: Performance comparison between different methods. Percentage improvements over the second-best baseline are shown in parentheses on a separate line. P@10=precision at 10, M@10=MRR@10, N@10=NDCG@10, R@10=recall at 10. (ECom data set has 10k queries with 5k candidates each—too large for RankZephyr-Itemwise and GPT-4o-mini to complete in 40h and many days respectively.)

models (7B vs 110M) struggle with quantity-constrained queries.

Ablation variant	MRR@10
DeepQuant	0.73
−Comparison regularization	0.723
−Supplementary Loss	0.63
−Number Reconstruction Loss	0.59
−Unit Prediction Loss	0.54

Table 4: Effect of removing number reconstruction loss and supplementary training for number comparisons.

4.5 Drill-down and ablation

4.5.1 Effect of various loss terms

Table 3 indicates that DeepQuant is highly effective compared against other baselines. In this section, we examine the effect of various losses we propose during finetuning. From Table 4 we observe that there is sharp decrease in MRR when the supplementary loss (Section 3.6) is removed. This reinforces the belief that sparsity in numeric signal can hurt training. We also observe that number reconstruction and unit prediction losses do contribute meaningfully to MRR@10. Lastly, we observe that comparison regularization (Eqn. 2) results in an additional modest gain.

4.5.2 Effect of quantity magnitudes

We compare the performance of DeepQuant vs. the best baseline QColBERT after binning the magnitude of quantities involved in the queries. For each bin, we show box-plots of the reciprocal rank (RR) distribution in Figure 5. We observe that DeepQuant performs better than QColBERT in several quantity magnitude buckets.

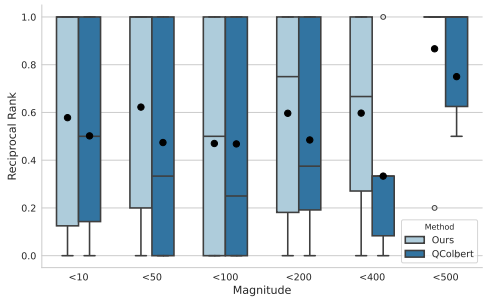

Figure 5: Performance variation across magnitude of quantities.

4.5.3 Effect of α

DeepQuant estimates α for each query. In Eqn. (10), if $\alpha \to 0$, that means quantities are more important for the query. Trusting this predicted α, we bucket queries into ranges of α and measure within-bucket MRR values for DeepQuant and the best baseline QColbert. Figure 6 shows that DeepQuant wins across α buckets, but the gap *is* larger for some low-α buckets, vindicating the design of DeepQuant.

4.5.4 Comparator-bucketed MRR

Figure 7 shows that DeepQuant is better than QColBERT, except on equality queries in FinQuant. The reason for this is that FinQuant (and MedQuant) require *exact* matching of quantities

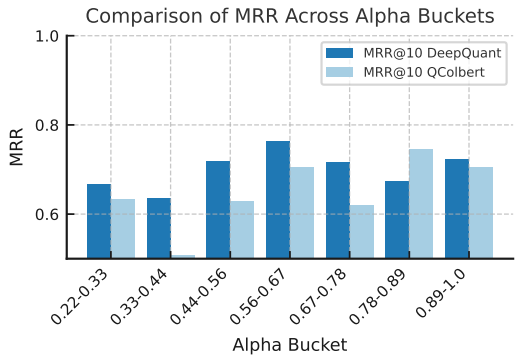

Figure 6: MRR in various α buckets.

for equality queries. Therefore for query `iPhone $500`, document `The price of iPhone 11 is $499` is judged as irrelevant. Experiments with a modified version of FinQuant where gold labels of equality queries were changed under human supervision are described in Appendix H.

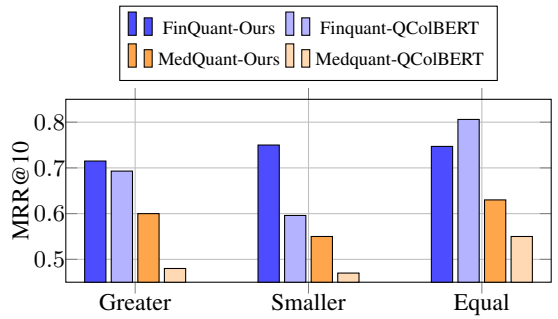

Figure 7: DeepQuant performance bucketed into $<, =, >$ query intents on FinQuant and MedQuant. We see that equality intents perform worst.

4.6 Proprietary data

The ECom data set is rich in equality constraints, but, unlike what baselines may infer, approximate (not exact) equality defines relevance. This leads to two outcomes: first, QBM25, QSPLADE and QColBERT perform very close to BM25, SPLADE and ColBERT respectively, and second, DeepQuant performs relatively better by soft equality scoring and unified contextualization. With 10k test queries and 5k candidate documents per query, RankZephyr, even in Itemwise mode, takes over 40 h; Listwise ordering is quite impractical. GPT-4o-mini is estimated to take many days. These latency limitations also establish the need to implement custom engineered retrieval networks rather than generic LLMs.

4.7 Anecdotes and analysis

For the query `iPhone 11 Pro Max below $1100`, DeepQuant correctly weighs the quantity $1149 in the passage `OnePlus 8 costs $599 , the OnePlus 7 T Pro costs $699 , the Samsung Galaxy S20 Ultra costs $1,199 and iPhone 11 Pro Max costs $1,149`, and finds it relevant. This is in contrast to QColBERT, which considers all quantities equally across the passage. However, for the query `video playback iPhone over 24 hours`, *all systems* judge the passage `iPhone 11 Pro will last four hours longer than the iPhone XS.` as irrelevant. This is because the query requires the models to know the battery life of `iphone XS` beforehand.

4.8 Multi-constraint queries

Our extensive experience with e-commerce search suggests that a vast majority of queries, even if they involve quantities rarely contain more than 1 constraint (specifically, 520 out of 43000 queries). Moreover, FinQuant and MedQuant contain no such queries. However as described earlier, unlike prior art (Almasian et al., 2024a; Ho et al., 2019, 2020), DeepQuant has no inherent limitation for retrieving such queries. Importantly, as described in Section 3.4, Eqn (1) and Section 3.9, Eqn (10), DeepQuant associates comparison intent with quantities, instead of the whole query like (Almasian et al., 2024a).

Our formulation is flexible in key ways. In particular, for multi-constraint queries, it is often the case that documents may partially violate some constraint, yet still remain overall relevant. Crucially, not all constraint violations are equally severe: some may be more tolerable than others depending on factors such as query intent, user behavior, or nature of gold documents. For example, in a query like `Smartphone under $500, storage 512GB`, a phone priced at $550 may still be highly relevant, but a phone with storage $128GB might be unacceptable. DeepQuant accommodates such cases by learning a soft, data-driven notion of constraint importance, rather than relying on rules or hand-crafted heuristics.

Moreover, we are able to train N_o end-to-end with distant supervision from $\langle Q, D_{\oplus}, D_{\ominus} \rangle$ without needing direct supervision of op. To test the effectiveness of DeepQuant on multi-constraint queries, we construct a synthetic dataset of such queries (2 and 3 constraints) from OpenFoodFacts.[3] This dataset consists of the nutrition breakdown of several food items. e.g., (banana, energy: 2243, nutrition-score: 14, etc.). To generate such

[3] https://github.com/openfoodfacts

Method	2-Comp		3-Comp	
	MRR@10	NDCG@10	MRR@10	NDCG@10
ColBERT	42.56	38.75	36.03	32.24
DeepQuant	47.71	41.91	41.24	36.02

Table 8: Performance on synthetic multi-constraint queries (2- and 3-attribute) from the OpenFoodFacts dataset.

Method	MRR@10	
	MSMARCO	FinQuant
ColBERT (trained on MSMARCO)	0.340	0.337[§]
DeepQuant (zero-shot)	0.210	0.734
DeepQuant (joint training)	0.332	0.731

Table 9: Evaluation on general-purpose (MSMARCO) and quantity-focused (FinQuant) retrieval. [§]Best baseline performance on FinQuant.

queries, we define a small set of multi-constraint query templates and populate them with relevant entries from the dataset. We construct a synthetic eval dataset of 10k queries and a training dataset of 50k positive pairs as documents which satisfy all the constraints present in the query. Results in Table 8 are clearly in favor of DeepQuant. We share the script used to generate the data.[4]

4.9 Impact on general retrieval

To assess whether DeepQuant adversely affects performance on general retrieval tasks without quantity expressions or intents, we evaluate it on the MSMARCO passage reranking benchmark, which contains 8.8 million passages of Web documents and approximately 500k real user queries from Bing. Crucially, we do not filter these queries for quantity comparison intent. We consider the following experimental settings:

ColBERT (trained on MSMARCO): ColBERT (Santhanam et al., 2022) trained from scratch on MSMARCO for 8 epochs with batch size 32 and learning rate 2×10^{-5}.

DeepQuant (zero-shot): DeepQuant initialized from the MSMARCO-trained ColBERT checkpoint, evaluated on MSMARCO without any further training.

DeepQuant (joint training): DeepQuant initialized from ColBERT and jointly trained on both MSMARCO and FinQuant datasets.

The results are shown in Table 9. The ColBERT model trained on MSMARCO achieves the best MRR@10 on that dataset, as expected. However, DeepQuant—trained jointly on general and quantity-focused data—achieves nearly the same performance on MSMARCO (0.332 vs. 0.340) while maintaining high performance on FinQuant (0.731 vs. 0.337). Expectedly, while DeepQuant exhibits lower zero-shot performance on MSMARCO (0.210), joint training on both FinQuant and MSMARCO yields an MRR@10 of 0.332—close to the ColBERT baseline (0.340). This demonstrates that the inductive biases introduced by DeepQuant's quantity-aware modules do not degrade retrieval quality on general-domain queries when trained jointly, validating its suitability for unified deployment across both numeric and non-numeric retrieval scenarios. These findings indicate that incorporating numerical reasoning modules into retrieval (via DeepQuant) does not compromise effectiveness on standard retrieval benchmarks when trained jointly. Moreover, in practical deployments, simple query intent classifiers can be used to selectively route quantity-based queries to DeepQuant, and all others to a standard retriever.

5 Conclusion

We introduced DeepQuant, a dense retrieval system that is tailored to queries with quantity comparison intent, as is overwhelmingly common in e-commerce and finance search. DeepQuant uses a PLM fine-tuned for reconstructing masked quantities with units, as well as end-to-end passage ranking proficiency. It infers the nature of quantity comparison, and deploys a novel passage scoring function that incorporates (uncertain) comparison operators. Experiments on three real-life datasets establish the promise of our approach.

[4] https://shorturl.at/oPBvJ

6 Limitations

Our work suggests several avenues for further exploration. Currently, we do not handle unit conversion, including equivalences between '12' and 'dozen', '3/4' and "three-fourth", etc. A relatively straight-forward way to extend DeepQuant would be to use an external tool to create ground truth and text-number conversions, and generalize masked quantity reconstruction to diverse units and surface forms. Our set-out goal excluded any computation, but integraring DeepQuant with some computation capability may be useful. E.g., the query `phone with 10h battery life` should score well with an item that specifies `screen-on time: 9 hours; standby time: 3 hours`. This is substantially more challenging, as we do not know at indexing time what form of arithmetic is needed on the document quantities. Extending beyond CQE as a single point of failure would make DeepQuant more robust. Toward that end, we may also want to enhance DeepQuant to deal with a distribution over possible quantity annotations. A more comprehensive evaluation with the most competitive LLMs like DeepSeek (DeepSeek-AI et al., 2025) will be of interest. Finally, we could extend from zero-shot to few-shot retrieval with LLMs.

References

Satya Almasian, Milena Bruseva, and Michael Gertz. 2024a. Numbers matter! bringing quantity-awareness to retrieval systems. *Preprint*, arXiv:2407.10283.

Satya Almasian, Vivian Kazakova, Philip Göldner, and Michael Gertz. 2023. CQE: a comprehensive quantity extractor. *arXiv preprint arXiv:2305.08853*.

Satya Almasian, Alexander Kosnac, and Michael Gertz. 2024b. QuantPlorer: Exploration of quantities in text. In *Advances in Information Retrieval*, pages 171–176, Cham. Springer Nature Switzerland.

Chung-Chi Chen, Hiroya Takamura, Ichiro Kobayashi, and Yusuke Miyao. 2023. Improving numeracy by input reframing and quantitative pre-finetuning task. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 69–77.

DeepSeek-AI, Daya Guo, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL Conference*.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE v2: Sparse lexical and expansion model for information retrieval. *Preprint*, arXiv:2109.10086.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal archive*, 12:23–38.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. *Preprint*, arXiv:2004.04487.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*.

Vinh Thinh Ho, Yusra Ibrahim, Koninika Pal, Klaus Berberich, and Gerhard Weikum. 2019. Qsearch: Answering quantity queries from text. In *ISWC*, pages 237–257. Springer.

Vinh Thinh Ho, Koninika Pal, Niko Kleer, Klaus Berberich, and Gerhard Weikum. 2020. Entities with quantities: extraction, search, and ranking. In *WSDM Conference (Demo)*, pages 833–836.

Vinh Thinh Ho, Koninika Pal, Simon Razniewski, Klaus Berberich, and Gerhard Weikum. 2021. Extracting contextualized quantity facts from web tables. In *WebConf*, pages 4033–4042.

Vinh Thinh Ho, Daria Stepanova, Dragan Milchevski, Jannik Strötgen, and Gerhard Weikum. 2022. Enhancing knowledge bases with quantity facts. In *WebConf*, pages 893–901.

Yusra Ibrahim, Mirek Riedewald, and Gerhard Weikum. 2016. Making sense of entities and quantities in web tables. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1703–1712.

Zhihua Jin, Xin Jiang, Xingbo Wang, Qun Liu, Yong Wang, Xiaozhe Ren, and Huamin Qu. 2021. Numgpt: Improving numeracy ability of generative pre-trained models. *Preprint*, arXiv:2109.03137.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its applications in retrieval. *Journal of Documentation*, 28(1):11–20.

Karen Sparck Jones, Steve Walker, and Stephen E Robertson. 2000. A probabilistic model of information retrieval: Development and comparative experiments (parts 1 and 2). *Information Processing and Management*, 36(6):779–840.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam M. Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *ArXiv*, abs/1602.02410.

George Katsogiannis-Meimarakis and Georgia Koutrika. 2023. A survey on deep learning approaches for text-to-sql. *The VLDB Journal*, 32:905–936.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Preprint*, arXiv:1910.13461.

Tongliang Li, Lei Fang, Jian-Guang Lou, Zhoujun Li, and Dongmei Zhang. 2021. Anasearch: extract, retrieve and visualize structured results from unstructured text for analytical queries. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 906–909.

Zhenwen Liang, Jipeng Zhang, Lei Wang, Wei Qin, Jie Shao, and Xiangliang Zhang. 2022. MWP-BERT: A numeracy-augmented pre-trained encoder for math word problems. In *NeurIPS Workshop on Math-AI*.

Tie-Yan Liu. 2009. Learning to rank for information retrieval. In *Foundations and Trends in Information Retrieval*, volume 3, pages 225–331. Now Publishers.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. 2024. Spreadsheetbench: Towards challenging real world spreadsheet manipulation. *Preprint*, arXiv:2406.14991.

Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836.

Bhaskar Mitra and Nick Craswell. 2018. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*, 13:1–126.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *Preprint*, arXiv:2312.02724.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.

Maciej Rybiński, Stephen Wan, Sarvnaz Karimi, Cécile Paris, Brian Jin, Neil Huth, Peter J. Thorburn, and Dean P. Holzworth. 2023. Sciharvester: Searching scientific documents for numerical values. *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. *NAACL*, abs/2112.01488.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Mandar Sharma, Rutuja Taware, Pravesh Koirala, Nikhil Muralidhar, and Naren Ramakrishnan. 2024. Laying anchors: Semantically priming numerals in language modeling. In *NAACL Findings*, pages 2653–2660.

Harsha Vardhan Simhadri, Ravishankar Krishnaswamy, Gopal Srinivasa, Suhas Jayaram Subramanya, Andrija Antonijevic, Dax Pryce, David Kaczynski, Shane Williams, Siddarth Gollapudi, Varun Sivashankar, Neel Karia, Aditi Singh, Shikhar Jaiswal, Neelam Mahapatro, Philip Adams, Bryan Tower, and Yash Patel. 2023. DiskANN: Graph-structured indices for scalable, fast, fresh and filtered approximate nearest neighbor search.

Jasivan Alex Sivakumar and Nafise Sadat Moosavi. 2025. How to leverage digit embeddings to represent numbers? In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7685–7697.

Daniel Spokoyny, Ivan Lee, Zhao Jin, and Taylor Berg-Kirkpatrick. 2021. Masked measurement prediction: Learning to jointly predict quantities and units from textual context. *Preprint*, arXiv:2112.08616.

Philip Sun, David Simcha, Dave Dopson, Ruiqi Guo, and Sanjiv Kumar. 2023. Soar: Improved indexing for approximate nearest neighbor search. In *Neural Information Processing Systems*.

Sowmya S Sundaram, Sairam Gurajada, Marco Fisichella, Deepak P, and Savitha Sam Abraham. 2022. Why are nlp models fumbling at elementary math? a survey of deep learning based word problem solvers. *Preprint*, arXiv:2205.15683.

Dhanasekar Sundararaman, Shijing Si, Vivek Subramanian, Guoyin Wang, Devamanyu Hazarika, and Lawrence Carin. 2020. Methods for numeracy-preserving word embeddings. In *EMNLP Conference*, pages 4742–4753, Online. Association for Computational Linguistics.

Avijit Thawani, Jay Pujara, Pedro A Szekely, and Filip Ilievski. 2021. Representing numbers in NLP: a survey and a vision. *arXiv preprint arXiv:2103.13136*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation learning with contrastive predictive coding. *Preprint*, arXiv:1807.03748.

Petar Veličković and Charles Blundell. 2021. Neural algorithmic reasoning. *Patterns*, 2(7):100273.

Petar Veličković, Matko Bošnjak, Thomas Kipf, Alexander Lerchner, Raia Hadsell, Razvan Pascanu, and Charles Blundell. 2022. Reasoning-modulated representations. *Preprint*, arXiv:2107.08881.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *EMNLP Conference*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Weixu Zhang, Yu Wang, and Ming Fan. 2023. Towards robustness of large language models on text-to-sql task: An adversarial and cross-domain investigation. In *Artificial Neural Networks and Machine Learning – ICANN 2023*, pages 181–192, Cham. Springer Nature Switzerland.

Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2024. Dense text retrieval based on pretrained language models: A survey. *ACM Trans. Inf. Syst.*, 42(4).

Dense Retrieval with Quantity Comparison Intent (Appendix)

A Mantissa encoding details

Suppose token and state vectors in the transformer have J elements each. This is partitioned into J_m elements for the mantissa and J_e for the exponent. Following (Jin et al., 2021) if the mantissa $m_b \in \mathbb{R}$ extracted from span b in Figure 1 is to be represented as $\vec{m}_b \in \mathbb{R}^{J_m}$, the jth element is the value of a Gaussian kernel with mean μ_j which depends on j:

$$\mu_j = \frac{10-(-10)}{J_m - 1} d + (-10), \quad \text{and} \tag{13}$$

$$\vec{m}_b[j] = \exp(-|m_b - \mu_j|^2/\sigma^2) \tag{14}$$

where we choose $\sigma = 1$ and J_m to be 90% of BERT embedding dimension $J = 768$.

B Fine tuning loss for numeral reconstruction

Consider the text fragment `iphone 15s 389 dollars`. The standard method that LMs use to encode text makes no distinction between words and quantities—they use byte-pair or wordpiece dictionaries to tokenize the text. Recent works (Thawani et al., 2021) suggest that this hinders appropriate representation of numbers. Moreover, these works show that using special methods for encoding numbers can improves the quality of representation significantly.

If the base embedding matrix $\boldsymbol{B}$ has J embedding dimensions for each token, we divide the J dimensions into two parts, J_e for the exponent and J_m for the mantissa, with $J_e + J_m = J$. Thus $\boldsymbol{y}[\texttt{[num]}, :J_e]$ and $\boldsymbol{y}[\texttt{[num]}, J_e:]$ are designated as the exponent and the mantissa embedding respectively, at the transformer output. Following Jin et al. (2021), we impose a MSE reconstruction loss $\mathcal{L}_{\text{man}}$ on the mantissa and a classification loss $\mathcal{L}_{\text{exp}}$ on the exponent, given ground truth quantity extractions from CQE (Almasian et al., 2023):

$$\mathcal{L}_{\text{num}} = \mathcal{L}_{\text{exp}} + \mathcal{L}_{\text{man}}, \tag{15}$$

where the exponent is limited to an integer $\in [-20, 20]$.

C Comparison networks N_{op}

The comparison scoring functions are designed with indexability in mind. For this reason we use three separate networks N_{op}, which score the belief "a op b" for op $\in \{<, =, >\}$.

$$N(\text{op}, \boldsymbol{y}_a, \boldsymbol{y}_b) = N_{\text{op},q}(\boldsymbol{y}_a) \cdot N_{\text{op},d}(\boldsymbol{y}_b) \tag{16}$$

where $N_{\text{op},q}, N_{\text{op},d} : \mathbb{R}^J \to \mathbb{R}^J$ are LRL feed-forward networks with 128 hidden units, that take $\boldsymbol{y}_a, \boldsymbol{y}_b$ as input. Avoiding early interaction and using dot-product at the output enables the use of approximate nearest neighbor search (Malkov and Yashunin, 2018; Guo et al., 2020; Sun et al., 2023; Simhadri et al., 2023). We use 2-layer feed-forward networks with each layer of width 128.

Query

`Samsung Galaxy Z less than $1000`

Sample $D_{\oplus}$

`For comparison, the OnePlus 7 T costs $549 , the Huawei P30 Pro costs £ 749 , the Samsung Galaxy S10+ costs £ 899 , the Samsung Galaxy Note 10+ costs $999 and the iPhone 11 Pro Max costs $1,149`

Sample $D_{\ominus}$

`The Galaxy Z Fold 2 was largely about earning our trust and proving that a $2,000 folding phone could work in the real world .`

Figure 10: A tricky win example.

D Retrieval fine-tuning loss details

We use the InfoNCE loss (van den Oord et al., 2019) for optimizing the retrieval objective. Given a query Q, a relevant document $D^{\oplus}$ and a sample of less relevant documents $\{D^{\ominus}\}$, we impose $\mathcal{L}_{\text{retr}} =$

$$-\log \frac{e^{\text{score}(D_{\oplus}|Q)/\tau}}{e^{\text{score}(D_{\oplus}|Q)/\tau} + \sum_{D_{\ominus}} e^{\text{score}(D_{\ominus}|Q)/\tau}}, \tag{17}$$

where $\text{score}(D_{\bullet}|Q)$ are computed as described in Section 3.9, and τ is the temperature, chosen as 0.02 based on validation.

E Example win/loss

Figure 10 shows an example of a non-trivial win for DeepQuant.

E.1 Bucketed query distributions

Figure 11 shows the distribution of queries among buckets defined by quantity magnitudes for the

MedQuant dataset Combined with Figure 5, this explains DeepQuant's lead over baselines. Figure 12 shows the distribution of queries among buckets defined by α as estimated by DeepQuant, for the FinQuant dataset. (Low α means quantities are important.) We see a concentration around the 0.5–0.75 range.

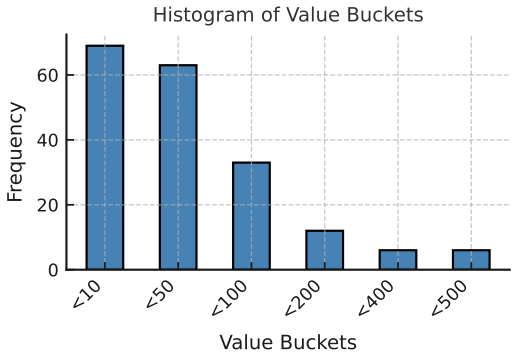

Figure 11: Distribution of queries bucketed by quantity magnitude as in Figure 5.

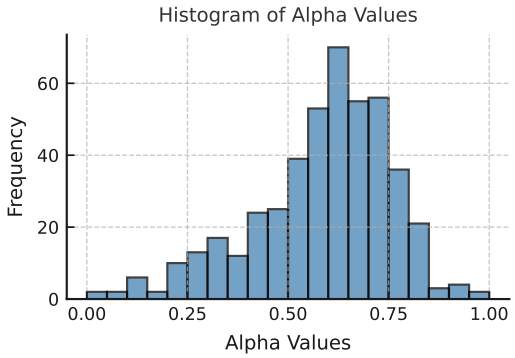

Figure 12: Distribution of queries bucketed by α as in Figure 6.

F Further details about baselines

BM25: BM25 (Jones et al., 2000) is a sparse vector space information retrieval model that ranks documents based on query terms frequency in each document, adjusted by the terms inverse document frequency and documents length.

ColBERT: ColBERT (Santhanam et al., 2022) captures fine-grained interactions between query and document tokens. It uses a `max-sim` late-interaction operator for improved retrieval accuracy over both sparse and bi-encoder dense retrievers by avoiding embedding the whole document into a single vector.

SPLADE: SPLADE (Formal et al., 2021) is a learned sparse retrieval model that projects text into high-dimensional space, producing sparse representations, by leveraging neural pre-trained models like BERT (Devlin et al., 2019) to predict term impacts. The additional terms are then used to expand query/documents for improved recall.

QBM25, QColBERT, QSplade: These (Almasian et al., 2024a) quantity aware variants of BM25, ColBERT, and SPLADE respectively, are designed to handle queries with numerical conditions. All these methods treat numbers separately from text, and score the numbers (without using any context) using handcrafted functions.

RankZephyr: RankZephyr (Pradeep et al., 2023) is a 7B-parameter, open-source LLM fine-tuned via instruction tuning with hard-negative mining for zero-shot Itemwise and Listwise (Liu, 2009) reranking. In Listwise mode, it is provided all candidate documents together, can exploit inter-document dependencies, and jointly reorders these document. In our experiments we had up to 5000 candidates which would be too long for the LLM's input context, so we processed these using a sliding window of 20 candidates at a time. In the faster Itemwise mode, the query and one document is submitted to the LLM at a time, and it outputs a per-document score. These scores are used for sorting the candidates.

GPT-4o-mini: We conduct all experiments with GPT-4o-mini in itemwise mode, since listwise ranking costs would be prohibitive. Unlike RankZephyr, which is finetuned with specific prompts, we found through trial-and-error the prompt in Figure 13 to be the most effective. We injected a single query-document pair into the appropriate placeholders in the prompt, and took the decoded token as our relevance score.

G Multi-comparator queries

Although we used simple single-constraint queries to simplify our exposition, DeepQuant naturally extends to multi-comparator queries such as `Laptop less than $500, storage more than 256GB and weight less than 2lb`. This is because, as per our formulation (Section 3.4), we associate a comparator with *each* quantity; see Eqn. (1). Our scoring function (3.9) works out of the box, given the summation is over $\{\!\{Q_{\#}\}\!\}$, which captures all the quantities present in the query. Moreover, since our method relies on distant supervision, we require no supervision on the comparators of associated quantities in queries.

System Prompt

```
You are a search engine relevance scorer. Your role is to evaluate the relevance of documents to a given query based on both semantic meaning and numeric constraints. You will consider multiple factors, including keyword overlap, semantic similarity, and numerical consistency, to determine a final relevance score. When processing a query, extract and interpret numeric values correctly, ensuring that greater/lesser comparisons and range constraints are properly respected. Your goal is to provide accurate, fair, and robust scoring that improves ranking quality in information retrieval tasks.
```

User prompt

```
Query: {query}, Document: {doc}. Rate relevance from 1 to 5. Respond with only a single number.
```

Figure 13: Prompt to GPT-4o-mini (Itemwise).

	Original	Curated
DeepQuant	73.2	81.2
QColBERT	80.1	81.3

Table 14: Performance after modified gold relevance labels for equality queries.

H Curated FinQuant benchmark

As we saw in Figure 7, DeepQuant's gains were smallest for equality queries in FinQuant. Closer manual scrutiny showed that gold labels were excessively restrictive in these instances. In many cases, strict equality was clearly not the query intent, and yet only passages with strictly equal quantities qualified as relevant.

For this reason, we introduce relaxed relevance labels for queries that involved equality constraint. We edited the labels via human supervision, if warranted. E.g., for query `Jeff Bezos worth 177 billion`, it is reasonable to match the document `Bezos, the founder of Amazon.com, is the worlds richest person with a US$ 176.6 billion fortune, according to the ranking`. However, we did not mark document `The central bank kept rates on hold at 1.00 percent, as expected.` as relevant for the query `interest rates of a bank =1.05 percentage`.

Table 14 shows that DeepQuant essentially matches QColBERT performance on equality queries once gold labels are sanitized with human supervision.