Dense Template Retrieval for Customer Support

Tiago Mesquita*
Zendesk
Lisbon, Portugal
tiago.mesquita@zendesk.com

Bruno Martins
INESC-ID & LUMLIS (Lisbon ELLIS Unit)
IST, University of Lisbon
Lisbon, Portugal
bruno.g.martins@inesc-id.pt

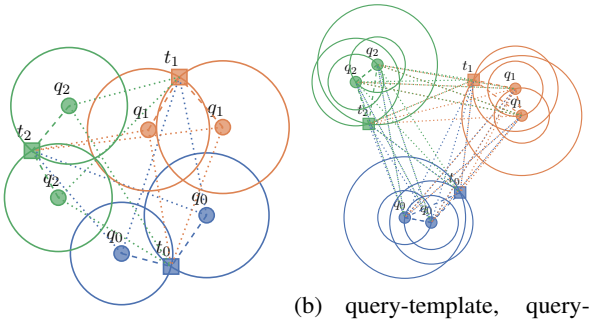
Mariana Almeida
Zendesk
Lisbon, Portugal
mariana.almeida@zendesk.com

Abstract

Templated answers are used extensively in customer support scenarios, providing an efficient way to cover a plethora of topics, with an easily maintainable collection of templates. However, the number of templates is often too high for an agent to manually search. Automatically suggesting the correct template for a given question can thus improve the service efficiency, reducing costs and leading to a better customer satisfaction. In this work, we propose a dense retrieval framework for the customer support scenario, adapting a standard in-batch negatives technique to support unpaired sampling of queries and templates. We also propose a novel loss that extends the typical query-centric similarity, exploiting other similarity relations in the training data. Experiments show that our approach achieves considerable improvements, in terms of performance and training speed, over more standard dense retrieval methods. This includes methods such as DPR, and also ablated versions of the proposed approach.

1 Introduction

A very common practice to make customer support more efficient is the use of templates for replies. The templates of replies are designed by customer support administrators to systematise the reply to frequent requests. Then, given a customer's request, an agent can pick a response from within a collection of predefined templates, this way saving time when replying to repetitive questions. Besides improving the throughput of human agents, the use of templates also assures uniformity in the handling of different customers, as requests with the same underlying problem should be handled with the same type of reply. However, customer support centers can have hundreds of templates, and finding the best template for a question is not an easy task, particularly for unexperienced agents. Automatically sorting and suggesting customer support templates (Bonatti et al., 2016; Yang and Kwok, 2012; Sneiders et al., 2016; Rei, 2019) can facilitate agent's work, reducing reply times, accelerating the learning curve of new agents, helping agents to focus on more added valued tasks, and overall providing a better customer support at reduced costs.



(a) query-template

(b) query-template, query-query and template-template

Figure 1: Illustration of 6 query representations $\{q_0, q_0, q_1, q_1, q_2, q_2\}$, together with respective template representations $\{t_0, t_1, t_2\}$ with t_i answering q_i, after enforcing different similarity relations. The distances between points represent dissimilarity, and dashed lines and circumferences represent similarity relations, whilst the dotted lines represent dissimilarity relations.

The recent advances in large pre-trained language models (Devlin et al., 2019; Vaswani et al., 2017), together with their successful use in question-answering (Karpukhin et al., 2020; Qu et al., 2021) and information retrieval (Xiong et al., 2021; Zhan et al., 2020, 2021; Yates et al., 2021), motivate the use of dense retrieval for template selection. Dense retrieval can be used to rank instances from the template collection, facilitating the selection of the correct template. Still, template ranking has specific characteristics when compared with more common retrieval scenarios in the literature: i) we have a strict many-to-one relation between queries and templates, in contrast to other common retrieval tasks (Nguyen et al., 2016); ii) the collection of templates is relatively small and

*This work was developed at Cleverly and Zendesk, under the context of Tiago Mesquita's Master of Science (M.Sc.) Thesis at IST, University of Lisbon.

Proceedings of the 29th International Conference on Computational Linguistics, pages 1106–1115
October 12–17, 2022.

generally in the order of hundreds, although also dynamically updated over time; iii) the length of the queries (emails with questions from customers) tends to be relatively long.

Given real-time and computation constraints in the template suggestion problem, we focus on bi-encoder models (Yates et al., 2021), which at prediction time only need to compute dense representations of queries and make fast comparisons with pre-computed representations of template candidates. We discard cross-encoder models that, despite often achieving higher retrieval performance (Yates et al., 2021), can have problems processing long queries and/or documents, and cannot take advantage of pre-computed template representations, involving higher computational costs that scale with the number of template candidates, and being often unsuitable for real-time applications. Still, our contributions are model independent, being also applicable to cross-encoders. Specifically, we make the following main research contributions:

- We compare different approaches in the task of template retrieval in customer support;
- We propose a new in-batch sampling strategy, that preserves the distributions of queries and templates to better select the information within batches, while exploring all possible query-template pairs in a batch;
- We propose a new loss function that exploits not only query-template similarity relations, but also query-query and template-template relations, yielding better representations for retrieval (see Figure 1).

Experiments with real-world customer support datasets show that both the in-batch sampling strategy and the expanded loss lead to improvements, in terms of template suggestion and training speed.

2 Related Work

Template retrieval for customer support has seen limited research in recent times. Most previous work has addressed the task as template classification with simple machine learning approaches (e.g., support vector machines or naïve Bayes) on top of representations obtained from bags-of-words (Bonatti et al., 2016; Yang and Kwok, 2012) or tailored pattern matching (Sneiders et al., 2016). A combination of retrieval and generative approaches is explored by Rei (2019), but without taking advantage of modern Transformer-based pre-trained language models. To the best of our knowledge, public literature on the topic has not explored recent advances in dense retrieval.

Most of the recent dense retrieval methods follow a BERT-based dual-encoder architecture and use a similarity function (e.g., cosine similarity) to produce ranking scores (Yates et al., 2021). The simplicity of the similarity function is crucial, allowing efficient retrieval through recent developments in Approximate Nearest Neighbour (ANN) search, such as FAISS (Johnson et al., 2019). These methods enable search speeds comparable with simpler sparse retrieval methods (e.g., BM25 (Robertson and Zaragoza, 2009)), whilst retaining better performance in most scenarios. Given the simplicity of dual-encoder architectures, most research has focused on improving the training procedure, namely by careful selection of the query-document pairs. For each query, the model should maximize similarity with all related documents (i.e., positives), whilst minimizing it for all unrelated documents (i.e., negatives). Most approaches have focused on the problem of selecting negative documents, usually falling under one of two categories: 1) efficient batching techniques, or 2) techniques that prioritize batch generation but generally sacrifice training efficiency.

In the first category, methods aim to maximize the amount of negatives available within the batches. The most efficient way to achieve this is by sharing negatives between all queries in the batch, an approach known as in-batch negatives (Karpukhin et al., 2020). More recent studies have proposed sharing negatives between multiple GPUs, allowing for massive batch sizes under parallel model training (Qu et al., 2021).

Although the previous techniques maximize the number of negatives seen during training, most of the instances are easily distinguishable, providing weak contributions to the loss function (i.e., easy negatives). Studies under the second category focus on the careful selection of negatives per query, looking for those that are useful for learning (i.e., hard negatives). The earliest approaches leveraged other retrieval methods to pool hard negatives, namely BM25 (Robertson and Zaragoza, 2009), picking highly ranked although irrelevant documents (Karpukhin et al., 2020; Xiong et al., 2021). Despite being effective at picking static

hard negatives, these approaches fail to adapt, as the model learns to rank the instances accordingly. ANCE (Xiong et al., 2021) addresses this issue by using the model itself to pool hard negative examples, dynamically adjusting the selection as the training progresses. In practice, dynamically generating and indexing embeddings for large document collections is costly, requiring separate GPUs to periodically maintain and refresh the index with prior checkpoints, whilst training in parallel. LTRe (Zhan et al., 2020) and later ADORE (Zhan et al., 2021) further refined these ideas, by freezing the weights of the document encoder and eliminating the need to refresh the index.

Although most approaches generally focus on a single category, some have tried to leverage techniques from both. DPR (Karpukhin et al., 2020) was one of the first studies to explore this idea, combining randomly pooled in-batch negatives with BM25 hard-negatives per query. Recently, STAR (Zhan et al., 2021) took the idea even further, by using static hard negatives pooled from the pre-trained model, but sharing them between all queries in the batch, similarly to in-batch negatives. Results showed that the combination provides an effective tool for stabilizing the biases introduced by the use of static negatives. Our technique also combines ideas from both categories, adapting in-batch negatives to the specific scenario of template retrieval, and performing hard negative sampling from all the in-batch negative examples.

Most previous studies on dense retrieval have generally also considered loss functions that enforce query-centric similarity relations, as these are explicitly related to the retrieval task. However, as shown in PAIR (Ren et al., 2021), models may benefit from exploring passage-centric similarity relations, potentially improving the representations. More recently, Li et al. (2021) proposed DANCE, which showed improvements by considering a loss function that combines query retrieval and document retrieval tasks. In our template retrieval scenario, these ideas are further explored, by taking into account all four possible relations between pairs of queries and templates.

3 Simple Dense Template Retrieval

We formally define the problem of template retrieval as follows: given a query q, the model must retrieve, from a collection of templates, the template t that better answers the query.

3.1 Architecture

Let us consider the commonly used dual-encoder architecture, as presented in DPR (Karpukhin et al., 2020), in which two independent encoders $E_Q(\cdot)$ and $E_T(\cdot)$ encode a query q and a template t into d-dimensional vectors, with different representation spaces. For ranking the templates, the cosine similarity between a query q and a template t can be computed from the respective representations:

$$\mathrm{s}(q,t) = \text{cosine-sim}\left(E_Q(q), E_T(t)\right). \quad (1)$$

3.2 Loss Function

The loss function for training the encoders should maximize the similarity between positive query-template pairs $s(q,t^+)$ and minimize the similarity between negative query-template pairs $s(q,t^-)$. A commonly used loss term for retrieval tasks is the negative log likelihood, comparing the positive template t^+ against a set of negative templates $\mathcal{T}^-$:

$$\mathcal{L}_q(q,t^+,\mathcal{T}^-) = -\log\left(\frac{e^{s(q,t^+)}}{e^{s(q,t^+)}+\sum_{t^-\in\mathcal{T}^-} e^{s(q,t^-)}}\right). \quad (2)$$

The final loss is then obtained by averaging the per-query loss from (2) over all queries (and template lists) considered in a batch from the dataset.

3.3 In-batch Negatives

Selecting good negative examples for training dense retrievers is still an open problem. Simple in-batch negatives, as described in DPR (Karpukhin et al., 2020), makes optimal use of the batch space, by sampling query-passage positive pairs and considering, for each query, all other passages within the batch as negatives. However, hidden in its simplicity lie two important assumptions: 1) the in-batch negatives are in fact negative passages; 2) the shared negatives provide a good estimation of instances within the full dataset. The weight of both assumptions is small for large corpora, where each document has a limited amount of related queries and vice-versa, making false in-batch negatives unlikely. Still, for smaller corpora such as those from customer support with templates, the assumptions can be problematic, requiring careful selection of the training pairs.

4 Improved Dense Template Retrieval

To improve on the method outlined in the previous section, we propose two innovative contributions.

The first relates to the in-batch sampling strategy, whilst the second refers to an expanded loss function that exploits different similarity relations for queries and templates.

4.1 Batch Generation

Template retrieval relates queries and templates in a strictly many-to-one correspondence, at the same time involving a small template collection. Moreover, since templates see different use, the number of queries per template varies considerably. These characteristics actively challenge the assumptions of vanilla in-batch negatives. In order to guarantee that the in-batch negatives are in fact negative, the sampled pairs must have different templates. This condition influences the distribution of training examples, penalizing frequently used templates and resulting in a distribution of negatives within the batch that does not follow the real data distribution.

4.1.1 Labeled In-batch Negatives

Given that each query has a single related template, labelling each text (i.e., query or template) in a training batch with the corresponding template identifier provides sufficient information to create all valid positive and negative pairs. More specifically, let t_i be the i-th template and $q_{i,j}$ be the j-th query from the sub-collection of queries that is answered by t_i. Given a batch of N_Q queries and N_T templates, with each text labeled with the corresponding template index i, we consider for each query $q_{i,j}$ the template t_i as positive, and all other templates t_n within the batch, with $n \neq i$, as negatives. This technique, wich we refer to as labeled in-batch negatives, not only prevents in-batch false negatives, but also eliminates the paired sampling restrictions (i.e., the training examples do not have to be explicit query-template pairs) imposed by vanilla in-batch negatives.

4.1.2 A Semi-independent Query-Template Sampling Strategy

As a general rule, we assume that training instances should follow 2 principles: 1) uniform sampling of positive pairs, since these offer explicitly labeled relevance information that should be uniformly explored; 2) sampling negatives according to a distribution that is consistent with the corpus. Vanilla in-batch negatives fails to follow both principles, as the distribution of negatives within the batch follows the distribution of templates available in the positive pairs, and not the real one. With labeled in-batch negatives, on the other hand, positives and negatives are not directly tied, enabling the consideration of both principles. To respect them, whilst maximizing the utility of the instances within the batch, we devised a semi-independent query-template sampling strategy, according to Algorithm 1 (and illustration in Figure 2).

Algorithm 1 Semi-independent query-template sampling procedure

Data: Let $E_{\mathcal{Q}}$ and $E_{\mathcal{T}}$ contain training queries and templates

Result: batch B, with b queries and b templates

1: Get set $\mathcal{T}$ with b templates uniformly selected from $E_{\mathcal{T}}$
2: Get set $\mathcal{Q}_{\mathcal{T}} \in E_{\mathcal{Q}}$ with the queries answered by $\mathcal{T}$
3: Get set $\mathcal{Q}$, by randomly sampling b queries from $\mathcal{Q}_{\mathcal{T}}$
4: Compose B, with the queries and templates in $\mathcal{Q}$ and $\mathcal{T}$

Note that unlike standard in-batch negatives, adapting this algorithm to support batches with a different number of templates and queries (i.e., batches of b_q queries and b_t templates, with $b_q \neq b_t \neq b$) is trivial. Still, we consider $b_q = b_t$ to be the most natural way to explore the information within a batch, as each query has a single positive template. This also mirrors the setup from the standard in-batch negatives approach.

4.2 Expanded Loss Function

We also propose a novel loss function that is expanded at the batch level, considering interactions not only between query-template pairs, but also query-query, template-template, and template-query pairs (see Figure 1). For that, let us first notice that each text in a batch (which can be either from a query or a template) is given a label corresponding to the correct template. The loss function for each batch can be defined with basis on the following generic loss term that takes two sets ($\mathcal{A}$ and $\mathcal{B}$) of labeled texts (that can be queries or templates) from a training batch:

$$\mathcal{L}(\mathcal{A},\mathcal{B}) = \\ -\frac{1}{|\mathcal{A}|}\sum_{i\in\mathcal{I},a_i\in\mathcal{A}_i}\frac{1}{|\mathcal{B}_i|}\sum_{b_i\in\mathcal{B}_i}\log\left(\frac{e^{s(a_i,b_i)}}{e^{s(a_i,b_i)}+\sum\limits_{j\neq i,b_j\in\mathcal{B}_j}e^{s(a_i,b_j)}}\right), \quad (3)$$

where $\mathcal{I}$ is the set of all labels in the batch, while $\mathcal{A}_i$ is the set of texts in $\mathcal{A}$ that have label i (i.e.,

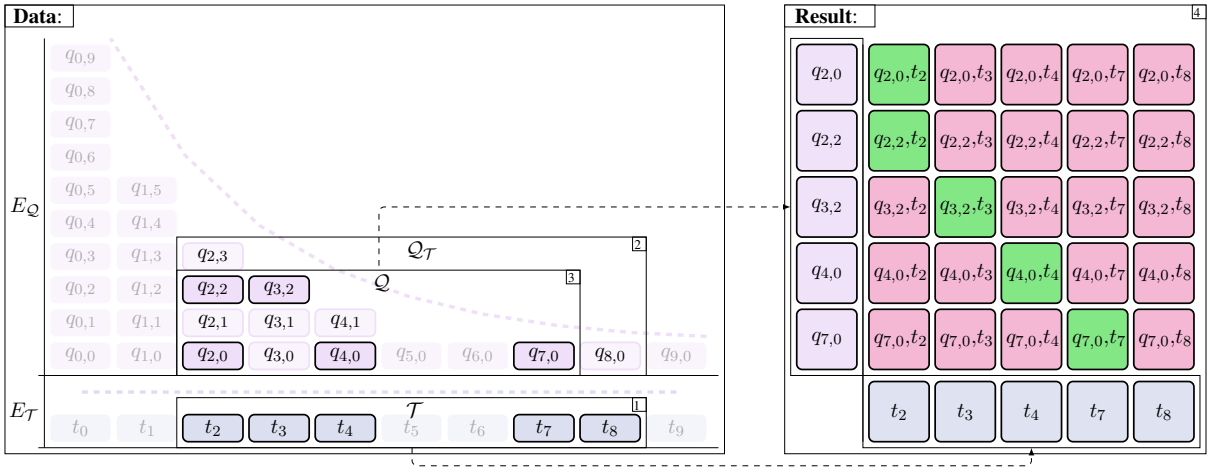


Figure 2: An illustrative case for the application of the semi-independent query-template sampling procedure (Algorithm 1) on an example training dataset. The illustrated collection contains templates $E_{\mathcal{T}}$ and queries $E_{\mathcal{Q}}$, where each query is represented above the template that answers it. Each step of the algorithm is represented with numbers and rectangles indicating the corresponding sets. The procedure results in the creation of batch B, and it is followed by the application of the labelled in-batch negatives technique, yielding the matrix of query-template pairs represented on the right. In the matrix, the positive pairs are shown in green, and the negative ones in red.

those that correspond to template i), and $\mathcal{B}_i$ is the set of texts in $\mathcal{B}$ that have label i.

4.2.1 Combining Different Loss Terms

The final loss of a batch is given by a weighted sum of four terms, each of them computed through (3):

$$\mathcal{L}_{\text{batch}} = \alpha\,\mathcal{L}(\mathcal{Q},\mathcal{T}) + \beta\,\mathcal{L}(\mathcal{Q},\mathcal{Q}) + \gamma\,\mathcal{L}(\mathcal{T},\mathcal{T}) + \theta\,\mathcal{L}(\mathcal{T},\mathcal{Q}), \quad (4)$$

where α, β, γ, and θ are adjustable hyper-parameters, and where the different loss terms correspond to different relations as follows:

1. $\mathcal{L}(\mathcal{Q},\mathcal{T})$; $\mathcal{A}=\mathcal{Q}$ is the set of all queries in a batch and $\mathcal{B}=\mathcal{T}$ is the set of all templates. This term corresponds to averaging the loss of each query $q \in \mathcal{Q}$, using the negative log likelihood of the positive template (2) combined with each possible negative template in the training batch;

2. $\mathcal{L}(\mathcal{T},\mathcal{T})$; $\mathcal{A}=\mathcal{T}$ and $\mathcal{B}=\mathcal{T}$ both correspond to the set of all templates in the batch. This term enforces the dissimilarity between distinct templates;

3. $\mathcal{L}(\mathcal{Q},\mathcal{Q})$; $\mathcal{A}=\mathcal{Q}$ and $\mathcal{B}=\mathcal{Q}$ both correspond to the set of all queries in the batch. This term enforces the dissimilarity between query representations from different templates, and promotes the similarity of representations for queries from the same template;

4. $\mathcal{L}(\mathcal{T},\mathcal{Q})$; $\mathcal{A}=\mathcal{T}$ is the set of all templates in a batch and $\mathcal{B}=\mathcal{Q}$ is the set of all queries in the batch. This term is the transpose of the first one, having a similar effect but acting on each template instead of each query;

4.2.2 In-batch Top-k Negatives

Section 2 discussed recent methods that use the model's own representations to guide the selection of hard-negatives (Xiong et al., 2021; Zhan et al., 2020, 2021). Although potentially effective, these techniques are computationally more demanding than the ones we propose, missing our efficiency goals. Still, inspired by these approaches, we consider a cheaper alternative of in-batch top-k negatives, that instead of retrieving the top-k negatives over the entire corpus, retrieves them from within the batch. By reusing the representations within a batch, this approach is much cheaper while also guaranteeing that the representations are synchronous. Unlike ANCE (Xiong et al., 2021) and the other methods, the value of selecting the in-batch top-k negatives is not on the selected hard negatives, since they are already present in the batch, but in discarding some instances. This delays over-fitting on simpler negatives, allowing the model to learn the harder ones.

5 Experiments

This section describes the experimental validation of the proposed contributions.

dataset	#queries			#t	$P_{80\%}$#tok		lang
	train	val	test		q	t	
CS-1	17858	3127	3918	445	113	396	en
CS-2	1092	187	650	82	111	164	pt

Table 1: Statistics for the two datasets. #t indicates the number of templates, and #tok indicates the text length in terms of multilingual DistilBERT$_{\text{base}}$ tokens.

5.1 Datasets and Metrics

We tested our approach on two private anonymized real-world datasets of email customer support interactions, in English and Portuguese.

The datasets are composed from a collection of real customer support interactions over email, where a human agent handpicked a template for answering a given query. The queries are user submitted and do not follow any particular formatting guidelines, ranging from typical emails (composed by a greeting, a body of text and a sign-off), to direct free-form questions. Templates of replies have more structure, following typical email conventions. Typically, these include trouble-shooting steps, clarifying information, notifications of hand-offs to other support agents/mediums, or a combination of these. The intents of the frequent requests that are answered with templates include login issues, password resets, after sale support, technical support, clarifications for products or promotions, complaints, recommendations, and identity verification, among others.

We split both datasets into 3 partitions, namely *train*, *val*, and *test*. The *test* split is composed of the most recent customer interactions, simulating the real temporal evaluation scenario, whilst the *train* and *val* splits are composed of the remaining examples on a 85/15 stratified split (see Table 1 for a characterization of the datasets).

The CS-1 and CS-2 datasets consider two different types of real world conditions. CS-1 has a relatively large size, and a large template collection, providing better training conditions and more representative results. On the other hand, CS-2 has a smaller, more specific, template collection, featuring non-English data (i.e., questions and templates in Portuguese) and fewer training examples.

5.1.1 Evaluation Metrics

We adopted the Recall@k (R@k) and Mean Reciprocal Rank (MRR) metrics for comparing models. MRR calculates the averaged reciprocal rank of the correct template, while R@k measures the ratio of queries in which the correct template is within the top-k. In particular, we track MRR@10 as a general indicator of ranking performance, and R@3 for matching the use-case of showing only the top-3 templates to agents.

Besides retrieval quality metrics, we also recorded the number of epochs involved in model fine-tuning, considering a early-stopping criterion based on MRR@10 over a validation split.

5.2 Experimental Setup

We now describe the approaches under comparison. Notice that we focused our analysis on pre-trained/fine-tuned multilingual models, envisioning real world scenarios where clients operate with different domains and languages.

5.2.1 Unsupervised Baselines

As a sparse retrieval baseline we consider a traditional BM25 (Lin et al., 2021) approach. For a dense retrieval baseline, we tested all multilingual models in Sentence-Transformers (Reimers and Gurevych, 2020), in a 0-shot manner, and report results for the best: `distiluse-base-multilingual-cased-v1`.

5.2.2 Fine-tuned Models

We tested a baseline approach based on randomly selected negative samples, similar to DPR (Karpukhin et al., 2020), as well as different ablated versions of our improved dense retrieval method. Both encoders on the dense retrievers (i.e., the query and template encoders) were initialized with the parameters of the `distiluse-base-multilingual-cased-v1` model from the Sentence-Transformers library (Reimers and Gurevych, 2020), as this was the best model in 0-shot retrieval experiments.

In more detail, and besides the complete proposed approach, we considered 5 other dense retrieval settings corresponding to the use of the vanilla loss ($\mathcal{L}(\mathcal{Q}, \mathcal{T})$) together with different mechanisms to construct the negative instances:

- **Random negatives**: randomly sample N negative templates for each query-template pair, as in the random sampling scheme described in DPR (Karpukhin et al., 2020);
- **In-batch neg^q**: sample B templates, weighed by frequency of positive queries and without

Methods	CS-1			CS-2		
	MRR@10	R@3	Epochs	MRR@10	R@3	Epochs
Unsupervised Baselines						
BM25	8.5 ± 0.0	10.1 ± 0.0	-	13.7 ± 0.0	16.6 ± 0.0	-
SBERT 0-shot	10.2 ± 0.0	12.0 ± 0.0	-	16.2 ± 0.0	20.2 ± 0.0	-
DPR (Karpukhin et al., 2020)						
Random negatives	39.4 ± 0.2	45.0 ± 0.4	25.0 ± 1.9	63.5 ± 1.9	73.6 ± 1.6	15.2 ± 2.2
Sampling Ablation						
In-batch neg^q	33.0 ± 0.8	37.9 ± 1.1	18.2 ± 6.0	61.6 ± 1.0	70.7 ± 1.2	20.2 ± 4.9
In-batch neg^t	38.0 ± 0.2	44.7 ± 0.4	22.2 ± 6.4	64.9 ± 1.6	75.0 ± 1.6	20.2 ± 6.4
Labeled in-batch neg^q	39.0 ± 0.6	45.2 ± 0.3	3.0 ± 0.7	63.2 ± 1.7	72.2 ± 1.0	10.5 ± 8.4
Labeled in-batch neg^{t,q}	$^{\ddagger}41.1 \pm 0.9$	$^{\ddagger}46.9 \pm 0.9$	4.5 ± 1.1	$^{\dagger}65.2 \pm 1.6$	$^{\dagger}75.4 \pm 1.2$	5.8 ± 0.8
Proposed approach	$^{\ddagger}\mathbf{42.2 \pm 0.3}$	$^{\ddagger}\mathbf{48.2 \pm 0.3}$	3.8 ± 0.8	$^{\ddagger}\mathbf{65.4 \pm 0.8}$	$^{\dagger}\mathbf{75.7 \pm 1.0}$	8.0 ± 1.6

Table 2: Results for both datasets, including mean and variance from 4 runs per model with different seeds. † and ‡ indicate significant improvements over the random negatives baseline, with p-values of 0.05 and 0.01, respectively, using the permutation test from Bassani (2022).

repetition, and a positive query for each of the sampled templates;

- **In-batch neg^t**: sample B templates, uniformly and without repetition, along with a positive query for each;

- **Labeled in-batch neg^q**: sample B queries, uniformly and without repetition, along with each respective template. If this produces repeated templates, we swap them with uniform samples not present in the batch;

- **Labeled in-batch neg^{t,q}**: corresponds to the complete version of the proposed sampling technique, as described in Section 4;

On what regards the hyper-parameters considered for model training, we used a batch-size $B = 32$ in all experiments with in-batch negatives, and $B = 8$ for the experiment with random negatives, with $N = 4$ negatives for each instance (i.e., sharing the negatives can improve space efficiency, and the choice of batch-size depended on the maximum capacity of the GPU used for the experiments, namely a NVIDIA Tesla T4 with 16GB of RAM). We used linear learning-rate scheduling with 500 warmup steps, and the ADAM optimizer (Kingma and Ba, 2015) with a learning-rate of 3e-5. We considered a maximum 30 epochs for the linear scheduler, stopping earlier if MRR@10 over the validation set stopped improving.

5.3 Experimental Results

Table 2 presents the obtained results, from which we can infer the following main conclusions:

1. The proposed sampling technique not only significantly outperforms all the alternative methods in both datasets, but it does so with considerably less training epochs. This result confirms the intuition that the common sampling strategies for IR fail to correctly model the one-to-many relation between user questions and templates.

2. The proposed loss, that also considers template-template and query-query similarity relations, improves the model further, yielding significant gains in terms of both the average performance and the corresponding variance. This result suggests that exploring semantic relations beyond the main ranking task is beneficial, likely being a result of learning more robust representations with better generalization capabilities, along with more stable training.

3. The poor performance of BM25 exposes the difficulty of the template retrieval task. Since each template covers a range of queries, the text is generally unspecific, resulting in reduced term overlap between templates and queries. Trained dense retrievers, on the other hand, were able to achieve good performance, showing that semantic relations are effectively superior to simple term matching.

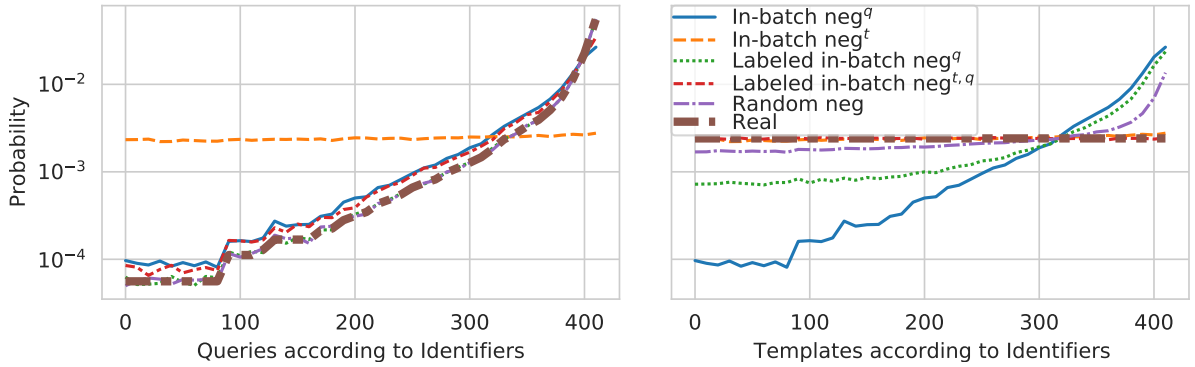


Figure 3: Comparison between the real and observed distributions, for the probability of occurrence of template identifiers in association to instances, obtained with different sampling techniques during model training, for queries (LEFT) and templates (RIGHT) on CS-1. The template identifiers associated to queries and templates are ordered by the real distribution and we plot the mean over bins of 10 identifiers, to reduce the number of data points and generate smoother lines that are easier to interpret.

4. The good results of the proposed techniques on CS-2 further validate our ideas, by proving the robustness to training conditions involving less data. It also validates the applicability of the proposed architecture in other languages. This is expected, as the approach simply refines the representation space of the base model. Given the multilingual pre-training, the approach is effectively able to transfer multilingual knowledge.

5.3.1 Analysis on the Sampling Techniques

The experiments reported in Table 2 also compared the different sampling techniques. To provide better insights over the practical differences between each method, we plotted the distributions of templates and queries, throughout training, for each technique. In order to do this, we recorded the template identifiers of the sampled queries and templates, at each step, for a total of 10 epochs in CS-1. The result is presented in Figure 3, which confirms the intuitions behind the design of the proposed sampling technique.

As expected, with vanilla in-batch negatives, queries and templates follow the same distributions, as they are sampled in pairs. This results in techniques that are only capable of optimizing the distribution of templates (i.e., **in-batch neg**^{t}) or queries (**in-batch neg**^{q}), but not both, resulting in sub-optimal performance. Labeled in-batch negatives are effectively able to decouple both distributions, fact that is key for providing good approximations for both templates and queries.

Labeled in-batch neg^{q} provides a good estimation over the distribution of queries, although the observed distribution of templates is slightly biased towards the most frequent. This results from the query-guided sampling technique, which can explain the slightly worse performance.

Labeled in-batch neg^{t,q}, on the other hand, is able to provide a uniform distribution of templates, whilst maintaining a distribution of queries very close to the real one. This provides, by far, the best global fit of both the distributions, resulting in the best overall performance.

The **random negatives** strategy, despite selecting templates on a per-query basis, is still slightly biased towards the most frequent templates, a result of the positive examples still following the query distribution. This, coupled with the reduced number of negatives, are likely the main factors for the lower performance.

Overall, the results seem to imply a correlation between the quality of the sampling techniques as an estimator of the involved distributions, and the observed retrieval performance of the strategy.

5.3.2 Analysis on the Loss Terms

The proposed loss function combines different terms, each enforcing a different similarity relation. In order to assess the contribution of each component, along with their interaction, we tested 5 different combinations of terms:

- $\mathcal{L}_1 = \mathcal{L}(\mathcal{Q}, \mathcal{T})$: control experiment, considering the negative log likelihood over the positive template;
- $\mathcal{L}_{2T} = \mathcal{L}(\mathcal{Q}, \mathcal{T}) + \mathcal{L}(\mathcal{T}, \mathcal{T})$: considers equal contribution of template-template and query-template relations;

Loss	W/o top-k neg		W/ top-k neg	
	MRR@10	R@3	MRR	R@3
$\mathcal{L}_1$	41.1 ± 0.9	46.9 ± 0.9	40.5 ± 1.4	47.1 ± 1.3
$\mathcal{L}_{2T}$	39.7 ± 0.9	45.8 ± 0.7	40.4 ± 0.5	46.8 ± 1.1
$\mathcal{L}_{2Q}$	‡**41.7** $\pm$ **0.5**	‡**47.7** $\pm$ **1.0**	$^{\ddagger}41.8 \pm 0.3$	$^{\ddagger}47.9 \pm 0.4$
$\mathcal{L}_3$	40.9 ± 0.6	46.8 ± 0.6	***42.2** $\pm$ **0.3**	***48.2** $\pm$ **0.3**
$\mathcal{L}_4$	41.4 ± 0.8	**47.6** $\pm$ **1.3**	41.5 ± 0.3	***48.3** $\pm$ **0.5**

Table 3: Ablation study on the terms of the loss and the in-batch top-k sampling, on the CS-1 dataset, including mean and variance intervals from 4 runs per model with different seeds. ‡ and * indicate significant improvements with p-values of 0.01 and 0.001, respectively, over the control experiment ($\mathcal{L}_1$, without top-k), according to the permutation test from Bassani (2022).

- $\mathcal{L}_{2Q} = \mathcal{L}(\mathcal{Q}, \mathcal{T}) + \mathcal{L}(\mathcal{Q}, \mathcal{Q})$: considers equal contribution of query-template and query-query relations;
- $\mathcal{L}_3 = \mathcal{L}(\mathcal{Q}, \mathcal{T}) + 0.5(\mathcal{L}(\mathcal{Q}, \mathcal{Q}) + \mathcal{L}(\mathcal{T}, \mathcal{T}))$: combines the query-template relations with equally contributing template-template and query-query relations;

For each of the considered losses, we also test the impact of using in-batch top-k negatives . We selected values for k experimentally, resulting in $k = 4$ for CS-1 and $k = 12$ for CS-2, with values below often leading to less stable training, and values above decreasing model performance. The results are presented in Table 3.

In agreement to what was reported in PAIR (Ren et al., 2021), the loss that combines query-template and template-template relations ($\mathcal{L}_{2T}$) is the one with lowest performance, suggesting some misalignment with the retrieval task. Combining query-template and query-query relations ($\mathcal{L}_{2Q}$) improves performance with respect to the control loss, and the combination of both ($\mathcal{L}_3$) improves it further, outperforming all others significantly, and suggesting complementarity of the two terms. It is important to note that the proposed loss function is effective without carefully tuning the contributions of each term (i.e., we only considered the configurations mentioned in Table 3), though tuning can perhaps improve performance even further.

The in-batch top-k sampling strategy improved performance and variance consistently, except for the simple query-template loss. This result suggests that the integration of the top-k sampling technique in the proposed loss function is beneficial, likely by regulating the contribution of each loss component.

6 Conclusions

This paper discussed challenges associated with retrieving templates for answering customer support questions, proposing a dense retrieval framework to address the task. This framework features innovative contributions in terms of (a) extending in-batch negatives to support unpaired sampling of queries and templates, and (b) a novel loss function that considers more similarity relations from the training data within each batch. Experiments on two different datasets of customer support interactions attest to the improvements brought forward by the proposed ideas. For future work, we plan to adapt and test the proposed techniques in other tasks that involve unbalanced corpora and large texts, such as general FAQ retrieval or question answering (Clark et al., 2020; De Bruyn et al., 2021).

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 873904. Bruno Martins was supported by Fundação para a Ciência e Tecnologia (FCT), through the INESC-ID multi-annual funding (i.e., the project with reference UIDB/50021/2020).

References

Elias Bassani. 2022. ranx: A blazing-fast Python library for ranking evaluation and comparison. In *Proceedings of the European Conference on Information Retrieval*.

Rogerio Bonatti, Arthur G. De Paula, Victor S. Lamarca, and Fabio G. Cozman. 2016. Effect of part-of-speech and lemmatization filtering in email classification for automatic reply. In *Proceedings of the Workshops at the AAAI Conference on Artificial Intelligence*.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8.

Maxime De Bruyn, Ehsan Lotfi, Jeska Buhmann, and Walter Daelemans. 2021. MFAQ: a multilingual FAQ dataset. In *Proceedings of the Workshop on Machine Reading for Question Answering*.

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

North American Chapter of the Association for Computational Linguistics: Human Language Technologies.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(03).

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yizhi Li, Zhenghao Liu, Chenyan Xiong, and Zhiyuan Liu. 2021. More robust dense retrieval with contrastive dual learning. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*.

Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS-MARCO: A human generated MAchine Reading COmprehension dataset. arXiv preprint arXiv:1611.09268.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Ricardo Rei. 2019. Towards a data-driven automation of customer support. Master's thesis, Instituto Superior Técnico of the University of Lisbon.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, QiaoQiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. PAIR: Leveraging passage-centric similarity relation for improving dense passage retrieval. In *Findings of the Association for Computational Linguistics*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4).

Eriks Sneiders, Jonas Sjöbergh, and Alyaa Alfalahi. 2016. Email answering by matching question and context-specific text patterns: Performance and error analysis. In *Proceedings of the World Conference on Information Systems and Technologies*. Springer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the International Conference on Neural Information Processing Systems*.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *Proceedings of the International Conference on Learning Representations*.

Weiwen Yang and Linchi Kwok. 2012. Improving the automatic email responding system for computer manufacturers via machine learning. In *Proceedings of the International Conference on Information Management, Innovation Management and Industrial Engineering*.

Andrew Yates, Rodrigo Nogueira, and Jimmy Lin. 2021. Pretrained transformers for text ranking: BERT and beyond. In *Proceedings of the ACM International Conference on Web Search and Data Mining*.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. Learning to retrieve: How to train a dense retrieval model effectively and efficiently. *arXiv preprint arXiv:2010.10469*.