Dense Template Retrieval for Customer Support

Tiago Mesquita

University of Lisbon, Instituto Superior Técnico and INESC-ID,
Av. Rovisco Pais 1049-001 Lisboa, Portugal,
tiago.mesquita.98@tecnico.ulisboa.pt

Abstract. Templated answers are used extensively in customer support scenarios, providing an efficient way to cover a plethora of topics, with an easily maintainable small collection of templates. Still, the number of templates is often too high for an agent to search. Automatically suggesting the correct template for a given question can improve the service efficiency, reducing costs and leading to a better customer satisfaction. In this work, we adapt the dense retrieval framework for the customer support scenario, modifying the commonly used 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 on private and public datasets show that our approach achieves considerable improvements in terms of performance and training speed.

1 Introduction

Customer support makes extensive use of templates for replies. Given a customer's query, an agent can pick a response from within a collection of predefined templates, with answers to common questions. Having a collection of templates saves time when replying to repetitive questions from customers. However, customer support centers can have hundreds of templates. Finding the best template for a question is not an easy task, particularly for unexperienced agents. Automatically sorting and suggesting customer support templates [1,21,18,13] 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 support at reduced costs.

The recent advances in large pre-trained language models [5,19], together with their successful use in question-answering [8,12] and information retrieval [20,24,23], motivates 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 [11]; ii) the collection of templates is relatively small and generally in the order of hundreds, although also dynamically updated over time; iii) the length of the queries (emails from costumers) tends to be long.

Given real-time and computation constrains in the template suggestion problem, this work focuses on bi-encoder models [22], which at prediction time only need to compute dense representations of queries and make a fast comparison with pre-computed representations of template candidates. We discard cross-encoder models that, despite often achieving higher retrieval performance[22] can have problems processing long queries and/or documents and cannot take advantage of pre-computed template representations. Despite that, our contributions are model independent, being also applicable to cross-encoders. We specifically make the following main research contributions:

- We compared classic and recent dense retrieval approaches in the task of template retrieval in customer support;
- We created and released[1] a corpus for template retrieval based on the *Customer Support on Twitter* dataset that is available on Kaggle[2], in order to motivate further research and benchmarking on the topic;
- We proposed 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 proposed 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.

Detailed experiments, using both public and real-life private 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 extracted representations, either from bags-of-words [1,21] or tailored pattern matching [18]. A combination of retrieval and generative approaches is explored in [13], but without taking advantage of 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 recent dense retrieval methods follow a BERT-based dual-encoder architecture and use a similarity function (e.g., cosine similarity) to produce ranking scores [22]. The simplicity of the similarity function is crucial, allowing efficient similarity searches by leveraging recent developments in Approximate Nearest Neighbour (ANN) retrieval, such as FAISS [7]. ANN enables search speeds comparable with simpler sparse retrieval methods (e.g., BM25 [17]), whilst retaining better performance in most scenarios. Given the simplicity of the architecture,

[1] We will release the corpus after paper acceptance.

[2] `https://www.kaggle.com/thoughtvector/customer-support-on-twitter`

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 negative selection problem and usually fall under one of two categories: (1) maximizing the quantity of negatives, trough efficient batching techniques or (2) maximizing the quality of negatives, prioritizing their selection but generally sacrificing 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 [8]. More recent studies have proposed sharing negatives between GPUs, allowing for massive batch sizes under parallel training [12].

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 [17] , picking highly ranked although irrelevant documents [8,20]. Despite being effective at picking static hard negatives, these approaches fail to adapt, as the model learns to rank the instances accordingly. ANCE [20] addresses this by using the model itself to pool hard negatives, 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 [24] and later ADORE [23] 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 [8] was one of the first studies to explore this idea, combining randomly pooled in-batch negatives with BM25 hard-negatives per query. Recently, STAR [23] 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 negatives.

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 [16], models may benefit from exploring passage-centric similarity relations, potentially improving the representations. In our template retrieval task, this technique has the potential to be even more effective. Since templates are purposely distinct, enforcing the distinctiveness of their representations can perhaps improve result quality.

Furthermore, since queries are related to a single template, queries sharing a template should have closer representations than queries related to different templates. Enforcing these relations with labeled in-batch negatives is trivial, as the labels provide the information needed to pair all related texts (i.e., queries and templates with the same label) and unrelated texts (i.e., different labels).

3 Simple Dense Template Retrieval

Following the customer support application introduced in Section 1, we formally define the problem of template retrieval as follows: given a query q, the model must retrieve the single template t, from a relatively small collection of N_t templates, that better answers the query.

Architecture: Let us consider the commonly used dual-encoder architecture, as presented in DPR [8], in which 2 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 is computed from the respective representations:

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

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 this retrieval task is the negative log likelihood comparing the positive template t^+ against a set of negative templates $\mathcal{T}^-$:

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

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

In-batch Negatives: Selecting negative examples for training dense retrievers is still an open problem, as seen in Section 2. Given the dynamic nature of customer support, we have chosen to focus on methods that maximize training efficiency. simple in-batch negatives, as described in DPR [8], makes optimal use of the batch space, by simply sampling query-passage positive pairs and considering, for each query, all other passages within the batch as negatives. However, hidden in its simplicity lie 2 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 pairs.

4 Improved Dense Template Retrieval

To improve on the method outlined in the previous section, we propose the two orthogonal innovative contributions that are described next. 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 follows the distribution of the templates, and not the real one.

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 correspond to the i-th template and $q_{i,j}$ to 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.

Semi-independent Query-Template Sampling: As a general rule, 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, as follows: For a batch size of b, we start by sampling b templates uniformly, in accordance with the second principle. From the set $\mathcal{T}$ of sampled templates, we compose the set $\mathcal{Q}_\mathcal{T}$ of all possible queries that are answered by templates included in $\mathcal{T}$. Finally, we produce the set $\mathcal{Q}$, by randomly sampling b queries from $\mathcal{Q}_\mathcal{T}$ and labeling them accordingly. This ensures that $\mathcal{Q}$ roughly follows the distribution of training examples, respecting the first principle.

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. 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 the batch:

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

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

Combining Different Loss Terms: The final loss of a batch is given by a weighted sum of four terms:

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

where α, β, γ and θ are adjustable hyper-parameters, and where the different loss terms are 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 (Equation 2) combined with each possible negative template in the 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 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;

In-batch Top-k Negatives: Section 2 discussed recent methods that use the model's own representations to guide the selection of hard-negatives [20,24,23]. Although potentially effective, these techniques are computationally more demanding than the ones we propose, missing our efficiency goals. 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 [20] 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 all others. This delays over-fitting on simpler negatives, allowing the model to learn the harder ones.

5 Experiments

This section presents the experimental validation of our contributions in two datasets of costumer support interactions.

5.1 Datasets and Metrics

The experiments were mainly conducted on a private dataset (CS-Private) consisting of real customer support interactions made over email, where a human agent handpicked templates for answering customer requests. Built from real interactions, this dataset is the most representative of the task at hand, although it cannot be made available due to company/client restrictions.

In the interest of reproducibility and to further stimulate research on the topic, we also crafted a dataset with public content[3] that corresponds to customer support interactions and approximates the task at hand. To create this public dataset (CS-Twitter), we built on previous work [6], in which the authors handcrafted a customer support dataset for chatbots, from the public *Customer Support on Twitter* data that is available on Kaggle[4]. We followed the same preprocessing, isolating the tweets related to Apple support and filtering out the noisy ones. We nonetheless processed the data further, to attend the needs of our problem. Given the apparent use of templated responses, we started by filtering out all tweets beyond the first interaction, as these were context specific and less likely to include templated answers. For the remaining tweets, we clustered similar responses, hopefully sharing a template, and assumed the clusters to be the golden template identifiers for the tweets in the clusters.

[3]The dataset will be made available after paper acceptance.

[4]`https://www.kaggle.com/thoughtvector/customer-support-on-twitter`

Dataset	#query-response pairs			#templates	$P_{80\%}$ token-length	
	train	val	test		queries	templates
CS-Twitter	7969	1425	2884	480	113	396
CS-Private	17858	3127	3918	415	34	35

Table 1: Statistics for CS-Twitter and CS-Private datasets. Token-length indicates the number of tokens of texts, in terms of DistilmBERT$_{\text{base}}$ tokens.

After analysing various text distances, we decided to capture the similarities between response tweets leveraging a pre-trained model from the Sentence-Transformers library [15] to produce sentence embeddings. In order to aggregate the similar sentences, we used HDBSCAN [2] to produce clusters over the response embeddings. Finally, we removed all clusters containing less than 5 elements and selected, for each cluster, the response that is closest to the centroid, as the template. From inspection of the produced data, HDBSCAN could provide better clusters than alternative methods, such as DBSCAN or k-means.

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 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).

Evaluation Metrics: We adopted the recall at k (R@k) and the Mean Reciprocal Rank (MRR) metrics to evaluate the rankings. Given that for each query a single template is correct, MRR calculates the averaged reciprocal rank of the correct template, while R@k measures the percentage of queries in which the top-k ranked templates contain the correct one.

5.2 Experimental Setup

Baselines: As a sparse retrieval baseline we consider a traditional BM25 [10] approach. For a dense retrieval baseline, we tested all multilingual models in Sentence-Transformers [14], in a 0-shot manner, and report results for the best: `distiluse-base-multilingual-cased-v1`. Finally, as an alternative to retrieval, and given the small size of the template space, we considered a simple multi-class classifier baseline, where each class corresponds to a template and we use the predicted probabilities as the ranking scores.

Pre-trained Language Models: Although our datasets were mostly in english, we only considered multilingual models in accordance with the real world scenario where we envision the models will be applied. Both encoders, on the trained dense retrievers, were initialized with the parameters of the `distiluse-base-multilingual-cased-v1` model from the Sentence-Transformers libraary [14], as this was the best model in 0-shot retrieval. For the classifier baseline, we used DistilmBERT$_{\text{base}}$ model.

Methods	CS-Private				CS-Twitter			
	MRR@10	R@3	R@10	Epochs	MRR@10	R@3	R@10	Epochs
Classifier	41.2	46.9	65.4	17	5.6	7.3	12.9	35
BM25	8.5	10.1	19.0	-	2.6	3.2	6.7	-
SBERT 0-shot	10.2	12.0	25.3	-	3.2	4.0	8.6	-
Random negatives	40.2	45.9	65.5	16	7.6	8.9	18.2	16
In-batch neg^t	38.2	44.8	63.8	24	6.6	7.7	14.7	44
In-batch neg^q	32.9	37.7	51.1	26	6.9	7.8	15.4	48
Labeled in-batch neg^q	39.2	45.6	63.7	4	7.2	8.4	16.7	6
Labeled in-batch $\text{neg}^{t,q}$	41.8	47.0	67.5	6	7.8	9.6	**19.1**	6
Proposed approach	**42.7**	**48.4**	**68.3**	3	**8.6**	**10.6**	18.9	8

Table 2: Experimental results on CS-Twitter and CS-Private.

Hyper-parameters: The batch-size (B) used for training was 32 for CS-Private, and 192 for CS-Twitter, in all experiments with in-batch negatives. For experiments with random negatives, we used B=8 in CS-Private and B=64, for CS-Twitter, sampling $N = 4$ negatives in both cases. We used larger batch-sizes for CS-twitter because the texts are much smaller then the emails in CS-Private. We also set the maximum number of training epochs to 30, for CS-Private, and 50 for CS-Twitter. Finally, we used linear learning-rate scheduling with 500 warmup steps, and the ADAM optimizer [9] with a learning-rate of 3e-5.

5.3 Experimental Results

Besides the baselines and proposed approach, we also considered 5 other settings corresponding to the use of the vanilla loss ($\mathcal{L}(\mathcal{Q}, \mathcal{T})$), but with different mechanisms to construct the negative instances:

- **Random negatives**: randomly samples N negative templates for each query-template pair;
- **In-batch neg^t**: samples B templates, uniformly and without repetition, along with a positive query for each template;
- **In-batch neg^q**: samples B templates, weighed by frequency of positive queries and without repetition, and a positive query for each template;
- **Labeled in-batch neg^q**: samples B queries, uniformly and without repetition, along with each positive template. If this produces repeated templates, we swap them with uniformly sampled templates not present in the batch;
- **Labeled in-batch $\text{neg}^{t,q}$**: corresponds to the proposed sampling technique, as described in Section 4;

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

1. The proposed sampling technique not only outperforms all the alternative methods in both datasets, but it does so with considerably less training. As expected, vanilla in-batch negatives is sub-optimal for template retrieval. The labeled in-batch negatives were key in overcoming the sampling limitations, and the proposed technique makes good use of its capabilities.
2. The proposed loss, that also considers template-template and query-query similarity relations, yields a significant performance boost. This result suggests that exploring semantic relations beyond the ranking task is beneficial, likely being a result of learning more robust representations with better generalization capabilities.
3. The overall poor performance on CS-Twitter exposes potential problems with the dataset, probably because it was created semi-automatically with reduced human supervision. Despite this, most results seem to be in agreement with those from CS-Private, enforcing the validity of the conclusions. In fact, the only noticeable discrepancy occurs in the experiment involving the **in-batch neg**^{q} strategy, with CS-Twitter exhibiting better relative performance. The discrepancy can be explained by the large batch-size (192), corresponding to almost half of the total number of templates. Since each batch will include a large number of sampled templates, the model is able to better explore the full corpus, independently of the sampling strategy.
4. 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, or the classification model, on the other hand, were able to achieve good performance, showing that semantic relations are effectively superior to simple term matching.
5. The classification baseline is in fact quite strong, outperforming several of the retrieval methods. This can be attributed to the small number of templates, although it should be emphasised that template collections can be highly dynamic in real settings, motivating the use of retrieval methods that can adapt to the collection without model re-training.

Analysis on the Sampling Techniques: The experiments in Table 2 already compare the different sampling techniques. To provide better insights over the practical differences of each method, we plot the distributions of templates and queries, throughout training, for each technique. To collect the data points, we record the template identifiers of the sampled queries and templates, at each step, for a total of 10 epochs in CS-Private. The result of this study is presented in Figure 1, which confirms the intuitions behind the design of the proposed sampling technique. As expected, vanilla in-batch negatives mirrors the distributions of queries and templates, 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, being key in providing good estimations. **Labeled**

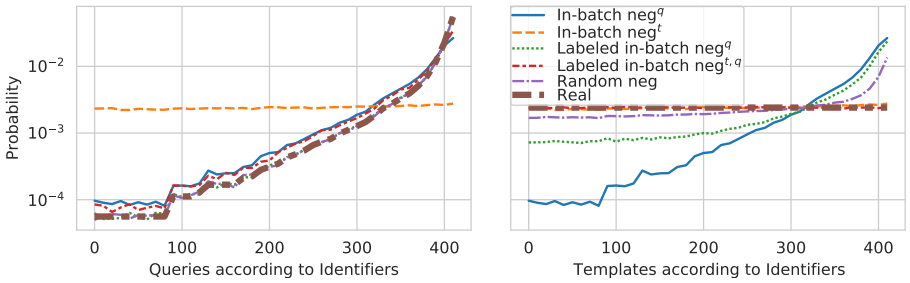

Fig. 1: Comparison between the real and observed distributions obtained with different sampling techniques, during training, for queries (LEFT) and templates (RIGHT). The template identifiers are ordered by the real distribution and we plot the mean over bins of 10 templates, to reduce the number of data points and generate smoother lines that are easier to interpret.

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 explains 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, providing by far, the best balance and accompanying best performance. **Random negatives**, 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 strong correlation between the quality of the sampling techniques, as an estimator of the involved distributions, and the observed retrieval performance.

Analysis on the Loss Terms: The proposed loss function combines different negative log-likelihoods, each enforcing a different similarity relation. In order to assess the contribution of each component, along with their interaction, we tested 5 different combinations:

- $\mathcal{L}(\mathcal{Q},\mathcal{T})$: corresponds to the control experiment and simply considers the common negative log likelihood over the positive template;
- $\mathcal{L}(\mathcal{Q},\mathcal{T}) + \mathcal{L}(\mathcal{T},\mathcal{T})$: considers equal contribution of and template-template and query-template relations,
- $\mathcal{L}(\mathcal{Q},\mathcal{T})+\mathcal{L}(\mathcal{Q},\mathcal{Q})$: considers equal contribution of query-template and query-query relations;
- $\mathcal{L}(\mathcal{Q},\mathcal{T})+0.5(\mathcal{L}(\mathcal{Q},\mathcal{Q})+\mathcal{L}(\mathcal{T},\mathcal{T})$: combines the query-template relations with equally contributing template-template and query-query relations;
- $\mathcal{L}(\mathcal{Q},\mathcal{T}) + \mathcal{L}(\mathcal{Q},\mathcal{Q}) + \mathcal{L}(\mathcal{T},\mathcal{T}) + \mathcal{L}(\mathcal{T},\mathcal{Q})$: similar to the previous loss, but additionally considering template-query relation.

Loss	in-batch top-k neg	CS-Private			CS-Twitter		
		MRR@10	R@3	R@10	MRR@10	R@3	R@10
$\mathcal{L}(\mathcal{Q},\mathcal{T})$	✗	41.8	47.0	67.5	7.8	9.6	**19.1**
	✓	41.9	48.3	67.9	8.1	9.6	18.4
$\mathcal{L}(\mathcal{Q},\mathcal{T})+\mathcal{L}(\mathcal{T},\mathcal{T})$	✗	38.7	45.3	65.4	7.2	8.7	18.0
	✓	39.7	45.9	65.6	7.6	8.8	18.0
$\mathcal{L}(\mathcal{Q},\mathcal{T})+\mathcal{L}(\mathcal{Q},\mathcal{Q})$	✗	41.5	46.7	67.8	8.0	9.6	17.8
	✓	41.6	**48.4**	67.1	7.7	9.2	18.6
$\mathcal{L}(\mathcal{Q},\mathcal{T})+0.5(\mathcal{L}(\mathcal{Q},\mathcal{Q})+\mathcal{L}(\mathcal{T},\mathcal{T}))$	✗	41.8	47.0	67.7	**8.6**	**10.6**	18.9
	✓	**42.7**	**48.4**	**68.3**	8.3	10.3	18.3
$\mathcal{L}(\mathcal{Q},\mathcal{T})+\mathcal{L}(\mathcal{Q},\mathcal{Q})+\mathcal{L}(\mathcal{T},\mathcal{T})+\mathcal{L}(\mathcal{T},\mathcal{Q})$	✗	41.3	47.0	65.9	8.4	10.2	19.0
	✓	41.4	47.7	67.4	7.5	8.7	16.8

Table 3: Ablation study on the components of the loss and in-batch top-k sampling, on CS-Twitter and CS-Private.

For each of the considered losses, we also test the impact of using in-batch top-k negatives. We selected values for k experimentally, after testing different powers of 2, resulting in $k = 4$ for CS-Private and $k = 24$ for CS-Twitter. The results of these experiments are presented in Table 3.

Just as reported in PAIR [16], the loss that combines query-template and template-template relations under-performs, suggesting some misalignment with the retrieval task. Similarly, combining query-template and query-query relations also appears ineffective. The combination of both, however, produces a considerable gains, suggesting complementarity. Moreover, in CS-Private, in-batch top-k sampling improved performance consistently, regardless of the considered loss. The same is however not true in the case of CS-Twitter. We believe this discrepancy relates to potential problems in the dataset. Specifically, during the construction of the dataset, the cases where the clustering algorithm failed to group responses sharing an underlying template produce incorrect hard negatives that are harmful for training, often being picked up in the top-k lists.

6 Conclusions

This paper discusses challenges associated with retrieving templates for answering customer support questions, proposing a dense retrieval framework to address the task. The proposed 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 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 [3,4].

References

1. Bonatti, R., De Paula, A.G., Lamarca, V.S., Cozman, F.G.: Effect of part-of-speech and lemmatization filtering in email classification for automatic reply. In: Workshops at the AAAI Conference on Artificial Intelligence (2016)
2. Campello, R.J.G.B., Moulavi, D., Sander, J.: Density-based clustering based on hierarchical density estimates. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds.) Advances in Knowledge Discovery and Data Mining (2013)
3. Clark, J.H., Choi, E., Collins, M., Garrette, D., Kwiatkowski, T., Nikolaev, V., Palomaki, J.: TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. Transactions of the Association for Computational Linguistics **8** (2020)
4. De Bruyn, M., Lotfi, E., Buhmann, J., Daelemans, W.: MFAQ: A multilingual FAQ dataset (2021)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North (2019)
6. Hardalov, M., Koychev, I., Nakov, P.: Towards automated customer support. In: International Conference on Artificial Intelligence: Methodology, Systems, and Applications. Springer (2018)
7. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with gpus. IEEE Transactions on Big Data (2019)
8. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., tau Yih, W.: Dense passage retrieval for open-domain question answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2020)
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2015)
10. Lin, J., Ma, X., Lin, S.C., Yang, J.H., Pradeep, R., Nogueira, R.: 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 (2021)
11. Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., Deng, L.: MS MARCO: A human generated machine reading comprehension dataset. In: CoCo@ NIPS (2016)
12. Qu, Y., Ding, Y., Liu, J., Liu, K., Ren, R., Zhao, W.X., Dong, D., Wu, H., Wang, H.: RocketQA: An optimized training approach to dense passage retrieval for open-domain question answering. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (2021)
13. Rei, R., Coheur, L., Graça, J.: Towards a Data-Driven Automation of Customer Support. Master's thesis, Instituto Superior Técnico of the University of Lisbon (2019)
14. Reimers, N., Gurevych, I.: Making monolingual sentence embeddings multilingual using knowledge distillation. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (2020)
15. Reimers, N., Gurevych, I., Reimers, N., Gurevych, I., Thakur, N., Reimers, N., Daxenberger, J., Gurevych, I., Reimers, N., Gurevych, I., et al.: 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 (2019)

16. Ren, R., Lv, S., Qu, Y., Liu, J., Zhao, W.X., She, Q., Wu, H., Wang, H., Wen, J.R.: PAIR: Leveraging passage-centric similarity relation for improving dense passage retrieval. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021 (2021)
17. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: Bm25 and beyond. Found. Trends Inf. Retr. **3**(4) (2009)
18. Sneiders, E., Sjöbergh, J., Alfalahi, A.: Email answering by matching question and context-specific text patterns: Performance and error analysis. In: New advances in information systems and technologies. Springer (2016)
19. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (2017)
20. Xiong, L., Xiong, C., Li, Y., Tang, K.F., Liu, J., Bennett, P.N., Ahmed, J., Overwijk, A.: Approximate nearest neighbor negative contrastive learning for dense text retrieval. In: International Conference on Learning Representations (2021)
21. Yang, W., Kwok, L.: Improving the automatic email responding system for computer manufacturers via machine learning. In: 2012 International Conference on Information Management, Innovation Management and Industrial Engineering. vol. 3. IEEE (2012)
22. Yates, A., Nogueira, R., Lin, J.: Pretrained transformers for text ranking: Bert and beyond. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining (2021)
23. Zhan, J., Mao, J., Liu, Y., Guo, J., Zhang, M., Ma, S.: Optimizing Dense Retrieval Model Training with Hard Negatives (2021)
24. Zhan, J., Mao, J., Liu, Y., Zhang, M., Ma, S.: Learning to retrieve: How to train a dense retrieval model effectively and efficiently. arXiv preprint arXiv:2010.10469 (2020)