Large Scale Question Paraphrase Retrieval with Smoothed Deep Metric Learning

Daniele Bonadiman
University of Trento
Trento, Italy
d.bonadiman@unitn.it

Anjishnu Kumar
Amazon Alexa
Seattle, USA
anjikum@amazon.com

Arpit Mittal
Amazon Alexa
Cambridge, UK
mitarpit@amazon.com

Abstract

The goal of a Question Paraphrase Retrieval (QPR) system is to retrieve similar questions that result in the same answer as the original question. Such a system can be used to understand and answer rare and noisy reformulations of common questions by mapping them to a set of canonical forms. This task has large-scale applications for community Question Answering (cQA) and open-domain spoken language question-answering systems. In this paper, we describe a new QPR system implemented as a Neural Information Retrieval (NIR) system consisting of a neural network sentence encoder and an approximate k-Nearest Neighbour index for efficient vector retrieval. We also describe our mechanism to generate an annotated dataset for question paraphrase retrieval experiments automatically from question-answer logs via distant supervision. We show that the standard loss function in NIR, triplet loss, does not perform well with noisy labels. We propose the smoothed deep metric loss (SDML), and with our experiments on two QPR datasets we show that it significantly outperforms triplet loss in the noisy label setting.

1 Introduction

In this paper, we propose a Question Paraphrase Retrieval (QPR) (Bernhard and Gurevych, 2008) system that can operate at industrial scale when trained on noisy training data that contains some number of false-negative samples. A QPR system retrieves a set of paraphrase questions for a given input, enabling existing question answering systems to answer rare formulations present in incoming questions. QPR finds natural applications in open-domain question answering systems, and is especially relevant to the community Question Answering (cQA) systems.

Open-domain QA systems provide answers to a user's questions with or without human intervention. These systems are employed by virtual assistants such as Alexa, Siri, Cortana and Google Assistant. Most virtual assistants use noisy channels, such as speech, to interact with users. Questions that are the output of an Automated Speech Recognition (ASR) system could contain errors such as truncations and misinterpretations. Transcription errors are more likely to occur for rarer or grammatically non-standard formulations of a question. For example 'Where Michael Jordan at?' could be a reformulation for 'Where is Michael Jordan?'. QPR systems mitigate the impact of this noise by identifying an answerable paraphrase of the noisy query and hence improves the overall performance of the system.

Another use of QPR is with cQA websites such as Quora or Yahoo Answers. These websites are platforms in which users interact by asking questions to the community and answering questions that have been posted by other users. The community-driven nature of these platforms leads to problems such as question duplication. Therefore, having a way to identify paraphrases can reduce clutter and improve the user experience. Question duplication can be prevented by presenting users a set of candidate paraphrase questions by retrieving them from the set of questions that have been already answered.

Despite some similarities, QPR task differs from the better known Paraphrase Identification (PI) task. In order to retrieve the most similar question to a new question, QPR system needs to compare the new question with all other questions in the dataset. Paraphrase Identification (Mihalcea et al., 2006; Islam and Inkpen, 2009; He et al., 2015) is a related task where the objective is to recognize whether a pair of sentences are paraphrases. The largest dataset for this task

Proceedings of the 2019 EMNLP Workshop W-NUT: The 5th Workshop on Noisy User-generated Text, pages 68–75
Hong Kong, Nov 4, 2019.

was released by Quora.com[1]. State-of-the-art approaches on this dataset use neural architectures with attention mechanisms across both the query and candidate questions. (Parikh et al., 2016; Wang et al., 2017; Devlin et al., 2019). However, these systems are increasingly impractical when scaled to millions of candidates as in the QPR setting, since they involve a quadratic number of vector comparisons per question pair, which are non-trivial to parallelize efficiently.

Information Retrieval (IR) systems have been very successful to operate at scale for such tasks. However, standard IR systems, such as BM25 (Robertson et al., 2004), are based on lexical overlap rather than on a deep semantic understanding of the questions (Robertson et al., 2009), making them unable to recognize paraphrases that lack significant lexical overlap. In recent years, the focus of the IR community has moved towards neural network-based systems that can provide a better representation of the object to be retrieved while maintaining the performance of the standard model. Neural representations can capture latent syntactic and semantic information from the text, overcoming the shortcomings of systems based purely on lexical information. Moreover, representations trained using a neural network can be task-specific, allowing them to encode domain-specific information that helps them outperform generic systems. The major components of a Neural Information Retrieval (NIR) system are a neural encoder and a k-Nearest Neighbour (kNN) index (Mitra and Craswell, 2017). The encoder is a neural network capable of transforming an input example, in our case a question, to a fixed size vector representation. In a standard setting, the encoder is trained via triplet loss (Schroff et al., 2015; Rao et al., 2016) to reduce the distance between a paraphrase vector when compared to a paraphrase vector with respect to a non-paraphrase vector. After being trained for this task, the encoder is used to embed the questions that can be later retrieved at inference time. The encoded questions are added to the kNN index for efficient retrieval. The input question is encoded and used as a query to the index, returning the top k most similar questions

Public datasets, such as Quora Question Pairs, are built to train and evaluate classifiers to identify paraphrases rather than evaluating retrieval systems. Additionally, the Quora dataset is not manually curated, thus resulting in a dataset that contains false-negative question paraphrases. This problem introduces noise in the training procedure when minimizing the triplet loss, since each question is compared with a positive and a negative example, that could be a false negative, at each training step. This noise is further exacerbated in approaches for training that exploit the concept of hard negatives, i.e., mining the non-paraphrase samples that are close to paraphrase samples in the vector space (Manmatha et al., 2017; Rao et al., 2016). Rather than treating these false negatives as a quirk of our data generation process, we recognize that false negatives are unavoidable in all large scale information retrieval scenarios with orders of millions or billions of documents - it is not feasible to get complete annotations as that would be of quadratic complexity in the number of documents. Usually, in these settings, randomly selected documents are treated as negative examples - thus the presence of noisy annotations with a bias towards false negatives is a recurring phenomenon in machine-learning based information retrieval.

In this work, we propose a loss function that minimizes the effect of false negatives in the training data. The proposed loss function trains the model to identify the valid paraphrase in a set of randomly sampled questions and uses label smoothing to assign some probability mass to negative examples, thus mitigating the impact of false negatives.

The proposed technique is evaluated on two datasets: a distantly supervised dataset of questions collected from a popular virtual assistant system, and a modified version of the Quora dataset that allows models to be evaluated in a retrieval setting. The effect of our proposed loss and the impact of the smoothing parameters are analyzed in Section 4.

2 Question Paraphrase Retrieval

In QPR the task is to retrieve a set of candidate paraphrases for a given query. Formally, given a new query q_{new}, the task is to retrieve k-questions, Q_k ($|Q_k| = k$), that are more likely to be paraphrases of the original question. The questions need to be retrieved from a given set of questions Q_{all} such that $Q_k \subseteq Q_{all}$, e.g., questions already answered in a cQA website.

[1]https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

2.1 System overview

The QPR system described in this paper is made of two core components: a neural encoder and an index. The encoder ϕ is a function ($\phi : Q \rightarrow \mathbb{R}^n$) that takes as input a question $q \in Q$ and maps it to a dense n-dimensional vector representation. The index is defined as the encoded set of all the questions that can be retrieved $\{\phi(q')|q' \in Q_{all}\}$ using the standard kNN search mechanism.

2.1.1 Encoder

The encoder ϕ used by our system is a neural network that transforms the input question to a fixed size vector representation. To this end, we use a convolutional encoder since it scales better (is easily parallelizable) compared to a recurrent neural network encoder and transformers (Vaswani et al., 2017), that have quadratic comparisons while maintaining good performance on sentence matching tasks (Yin et al., 2017). Additionally, convolutional encoders are less sensitive to the global structure of the sentence then recurrent neural network thus being more resilient to noisy nature of user-generated text The encoder uses a three-step process:

1. An embedding layer maps each word w_i in the question q to its corresponding word embedding $x_i \in \mathbb{R}^{e_{dim}}$ and thereby generating a sentence matrix $X_q \in \mathbb{R}^{l \times e_{dim}}$, where l is number of words in the question. We also use the hashing trick of (Weinberger et al., 2009) to map rare words to m bins via random projection to reduce the number of false matches at the retrieval time.

2. A convolutional layer (Kim, 2014) takes the question embedding matrix X_q as input and applies a trained convolutional filter $W \in \mathbb{R}^{e_{dim} win}$ iteratively by taking at each timestep i a set of win word embeddings. This results in the output:

$$h_i^{win} = \sigma(W x_{i-\frac{win}{2}:i+\frac{win}{2}} + b) \quad (1)$$

   , where σ is a non linearity function, $tanh$ in our case, and $b \in \mathbb{R}$ is the bias parameter. By iterating over the whole sentence it produces a feature map $\mathbf{h}^{win} = [h_1^{win}, .., h_l^{win}]$.

3. A global max pooling operation is applied over the feature map ($\hat{h}^{win} = max(\mathbf{h}^{win})$) to reduce it into a single feature value. The convolutional and global max pooling steps described above are applied multiple times (c_{dim} times) with varying window size with resultant $\hat{h}$ values concatenated to get a feature vector $h \in \mathbb{R}^{c_{dim}}$ which is then linearly projected to an n-dimensional output vector using a learned weight matrix $W_p \in \mathbb{R}^{n \times c_{dim}}$.

2.1.2 kNN Index

Despite there is no restriction on the type of kNN index that can be used, for performance reasons, we use FAISS[2] (Johnson et al., 2017) as an approximate kNN index[3]. All the questions (Q_{all}) are encoded *offline* using the encoder ϕ and added to the index. At retrieval time a new question is encoded and used as a query to the index. The kNN index uses a predefined distance function (e.g. Euclidean distance) to retrieve the nearest questions in the vector space.

3 Training

Typical approaches for training the encoder use triplet loss (Schroff et al., 2015; Rao et al., 2016). This loss attempts to minimize the distance between positive examples while maximizing the distance between positive and negative examples.

The loss is formalized as follows:

$$\sum_i^N [\|\phi(q_i^a) - \phi(q_i^p)\|_2^2 - \|\phi(q_i^a) - \phi(q_i^n)\|_2^2 + \alpha]_+ \quad (2)$$

where q_i^a is a positive (anchor) question, q_i^p is a positive match to the anchor (a valid paraphrase), q_i^n is a negative match (i.e. a non-paraphrase), α is a margin parameter and N is the batch size.

In a recent work by Manmatha et al. 2017 the authors found that better results could be obtained by training the above objective with hard negative samples. These hard negatives are samples from the negative class that are the closest in vector space to the positive samples, hence most likely to be misclassified.

However, in our case, and in other cases with noisy training data, this technique negatively impacts the performance of the model since it starts focusing disproportionately on any false-negative samples in the data (i.e. positive examples labelled

[2] https://github.com/facebookresearch/faiss

[3] FAISS provides efficient implementations of various approximated kNN search algorithms for both CPU and GPU

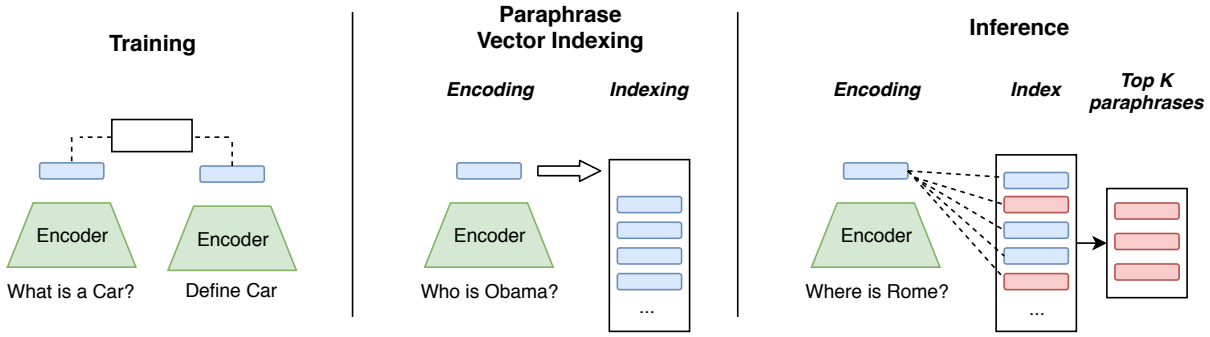

Figure 1: System

as negative due to noise) making the learning process faulty. For example in the Quora dataset positive examples are marked as paraphrase, duplicate, by users using the website however there is no manual check for the negative examples, thus leading to a number of false negatives that happens to be close in the vector space.

3.1 Smoothed Deep Metric Learning

In this paper, we propose a new loss function that overcomes the limitation of triplet loss in the noisy setting. Instead of minimizing the distance between positive examples with respect to negative examples, we view the problem as a classification problem. Ideally, we would like to classify the paraphrases of the original question amongst all other questions in the dataset. This process is infeasible due to time and memory constraints. We can, however, approximate this general loss by identifying a valid paraphrase in a set of randomly sampled questions (Kannan et al., 2016). We map vector distances into probabilities similar to Goldberger et al. 2005 by applying a softmax operation over the negative squared euclidean distance:

$$\hat{p}(a,i) = \frac{e^{-\|\phi(q^a)-\phi(q^i)\|_2^2}}{\sum_j^N e^{-\|\phi(q^a)-\phi(q^j)\|_2^2}} \quad (3)$$

where q^a is an anchor question and q^j and q^i are questions belonging in a batch of size N containing one paraphrase and $N-1$ randomly sampled non-paraphrases. The network is then trained to assign a higher probability, hence a shorter distance, to pair of questions that are paraphrases.

Additionally, we apply the label smoothing regularization technique (Szegedy et al., 2016) to reduce impact of false negatives. This technique reduces the probability of the ground truth by a smoothing factor ϵ and redistributes it uniformly across all other values, i.e.,

$$p'(k|a) = (1-\epsilon)p(k|a) + \frac{\epsilon}{N} \quad (4)$$

where $p(k|a)$ is the probability for the gold label. The new smoothed labels computed in this way are used to train the network using Cross-Entropy (CE) or Kullback-Leibler (KL) divergence loss[4]. In our setting, the standard cross-entropy loss tries to enforce the euclidean distance between all random points to become infinity, which may not be feasible and could lead to noisy training and slow convergence. Instead, assigning a constant probability to random interactions tries to position random points onto the surface of a hypersphere around the anchor which simplifies the learning problem.

The sampling required for this formulation can be easily implemented in frameworks like PyTorch (Paszke et al., 2017) or MxNet (Chen et al., 2015) using a batch of positive pairs $< q_{1,j}, q_{2,j} >$ derived from a shuffled dataset, as depicted in Figure 2. In this setting, each question $q_{1,i}$ would have exactly one paraphrase, i.e., $q_{2,i}$ and $N-1$ all other questions $q_{2,j}$ when $j \neq i$ would serve as counter-examples. This batched implementation reduces training time and makes sampling tractable by avoiding sampling N questions for each example, reducing the number of forward passes required to encode the questions in a batch from $\mathcal{O}(N^2)$ in a naive implementation to $\mathcal{O}(2N)$.

[4]In this setting, CE loss and KL divergence loss are equivalent in expected values. However, we use the KL divergence loss for performance reasons.

Figure 2: Batched implementation of the loss with smoothing parameter $\epsilon = 0.3$ and batch size $N = 3$. Each paraphrase pair $< q_{1,j}, q_{2,j} >$ in the batch is compared with all the others questions in the batch.

4 Experiments

In this section, we present the experimental setup used to validate our approach for QPR using the Smoothed Deep Metric Learning (SDML) technique.

4.1 Datasets

In order to generate a dataset for question paraphrase retrieval, we propose a technique that uses distant supervision to create it automatically from high-precision question-answer (QA) logs. Additionally, due to the proprietary nature of our internal dataset, we tested our approach on a modified version of the Quora paraphrase identification dataset that has been adapted for the paraphrase retrieval task.

4.1.1 Open Domain QA dataset

Our open domain Q&A dataset is created by weak supervision method using high precision QA logs of a large scale industrial virtual assistant. From the logs, we retrieve 'clusters' of questions that are mapped to the same answer. However, we notice that this may generate clusters where unrelated questions are mapped to a generic answer. For instance, many different math questions may map to the same answer; e.g. a given number. To further refine these clusters, the data is filtered using a heuristic based on an intra-cluster similarity metric that we call cluster *coherence*, denoted as c. We define this metric as the mean Jaccard similarity (Levandowsky and Winter, 1971) of each question in a cluster to the cluster taken as the whole.

Mathematically, for a given cluster $\mathbb{A} = \{q_1, q_2 ... q_n\}$ and defining $\mathbb{T}_{q_i} = \{w_{i_1}, w_{i_2}, ... w_{i_k}\}$ as shorthand for the set of unique tokens present in a given question, the coherence of the cluster is defined as:

$$\mathbb{S} = \bigcup_{i=1}^{n} \mathbb{T}_{q_i} \tag{5}$$

$$c = \frac{1}{n} \Sigma_{i=1}^{n} \frac{|\mathbb{T}_{q_i} \cap \mathbb{S}|}{|\mathbb{S}|} \tag{6}$$

In practice, we found that even a small coherence filter ($c < 0.1$) can eliminate all incoherent question clusters. Our approach to weak supervision can be considered as a generalized instance of the candidate-generation noise-removal pipeline paradigm used by Kim et al. 2018. Once the incoherent clusters are removed from the dataset, the remaining clusters are randomly split in an 80:10:10 ratio into training, validation and test sets and question pairs are generated from them[5]. A second filter is applied to remove questions in the validation and test sets that overlap with questions in the training set. The final output of the weak supervision process is a set of silver labelled clusters with $> 99\%$ accuracy based on spot-checking, a random sample of 200 clusters.

4.1.2 Quora dataset

We introduce a variant of the Quora dataset for QPR task. The original dataset consists of pairs of questions with a positive label if they are paraphrases, and a negative label if they are not. Similarly to Haponchyk et al. (2018), we identify question clusters in the dataset by exploiting the transitive property of the paraphrase relation in the original pairs, i.e., if q_1 and q_2 are paraphrases, and q_2 and q_3 are paraphrases then q_1 and q_3 are also paraphrases, hence q_1, q_2, and q_3 belong to the same cluster. After iterating over the entire dataset, we identified $60,312$ question clusters. The question clusters are split into the training, validation and test sets such that the resulting validation and test set contains roughly $5,000$ question pairs each, and the training set contains $219,369$ question pairs[6]. The kNN index is composed of all the questions in the original Quora datasets (including questions that appear only as negative, thus not being part of any cluster) for a total of $556,107$ questions.

[5]The open-domain QA dataset contains on order of 100k - 1M training clusters, 10k - 100k clusters each for validation and testing, and a search index of size $\approx 10M$.

[6]The code to generate the splits will be released upon acceptance.

4.2 Experimental setup

We described the architecture of our encoder previously in section 2.1.1. For experimentation, we randomly initialized word embeddings. The size of vocabulary for Quora dataset is fixed at 50,000 whereas for the bigger open-domain QA dataset we used a vocabulary of size 100,000. To map rare words we use the hashing trick (Weinberger et al., 2009) with 5,000 bins for the Quora dataset and 10,000 bins for the QA dataset.

We set the dimensionality of word embeddings at 300 (i.e., $e_{dim} = 300$); the convolutional layer uses a window size of 5 (i.e., $win = 5$) and the encoder outputs a vector of size $n = 300$. For triplet loss the network is trained with margin $\alpha = 0.5$. The default batch size for all the experiments is 512 (i.e., $N = 512$) and the smoothing factor for SDML, ϵ, is 0.3. For all experiments training is performed using the Adam optimizer with learning rate $\lambda = 0.001$ until the model stops improving on the validation test, using early stopping (Prechelt, 1998) on the ROC AUC metric (Bradley, 1997).

4.3 Evaluation

We use *IVF2000, Flat* configuration of the FAISS library as our index, which is a hierarchical index consisting of an index of k-means centroids as the top-level index. For evaluation, we retrieve 20 questions with 10 probes into the index each returning a pair of paraphrase questions, with an average query time of < 10 ms. These questions are used to measure the system performance via standard information retrieval metrics, Hits@N ($H@N$) and Mean Reciprocal Rank (MRR). $H@N$ measures if at least one question in the first N that are retrieved is a paraphrase and MRR is the mean reciprocal rank (position) at which the first retrieved paraphrase is encountered.

4.4 Results

In the first set of experiments, we measured the impact of varying the smoothing factor ϵ. The results for the Quora validation set are presented in Table 1. We observe that the presence of smoothing leads to a significant increase over the baseline (simple cross-entropy loss) and increasing this parameter has a positive impact up to $\epsilon = 0.3$.

In our second experiment, we hold the ϵ constant at 0.3 and experiment with varying the number of negative samples. Table 2 shows the effect of an increase in the number of negative examples in a batch. The model's performance reaches its maximum value at $N = 512$, i.e., with 511 negative samples for each positive sample. We want to point out that we limited our exploration to 1024 due to memory constraints. However, better performance may be achieved by further increasing the number of examples, since the batch becomes a better approximation of the real distribution.

ϵ	H@1	H@10	MRR
0	0.5568	0.7381	0.6217
0.1	0.5901	0.7841	0.6591
0.2	0.6030	0.8090	0.6762
0.3	**0.6133**	0.8113	**0.6837**
0.4	0.6107	**0.8144**	0.6815

Table 1: Impact of smoothing factor ϵ on the Quora validation set.

N	H@1	H@10	MRR
32	0.5389	0.7444	0.6103
64	0.5710	0.7726	0.6410
128	0.6093	0.8085	0.6777
256	0.6112	**0.8141**	0.6833
512	**0.6133**	0.8113	**0.6837**
1024	0.6081	0.8008	0.6764

Table 2: Impact of the batch size N on the Quora validation set. For computing SDML a batch consists of a paraphrase and $N - 1$ negative examples.

Table 3 and 4 compare the proposed loss with the triplet loss with random sampling, TL(Rand). We compared the proposed approach with two variants of triplet loss that uses different distance functions Euclidean Distance (EUC) and Sum of Squared Differences (SSD). The Euclidean distance is the standard distance function for triplet loss implementation present in popular deep learning frameworks, PyTorch and Mxnet, whereas SSD is the distance function used in the original paper of Schroff et al. 2015. Our approach improves over the original triplet loss considerably on both datasets. The SSD distance also outperforms the EUC implementation of the loss.

Tables 5 and 6 show the results on the open domain QA dataset validation and test set. TL(Rand) is the triplet loss with random sampling of negative examples, whereas TL(Hard) is a variant with hard negative mining. In both cases, the SDML outperforms triplet loss by a considerable mar-

Loss	Dist	H@1	H@10	MRR
TL (Rand)	EUC	0.4742	0.6509	0.5359
TL (Rand)	SSD	0.5763	0.7640	0.6421
SDML	SSD	**0.6133**	**0.8113**	**0.6837**

Table 3: Comparison of different loss functions on Quora validation set.

Loss	Dist	H@1	H@10	MRR
TL (Rand)	EUC	0.4641	0.6523	0.5297
TL (Rand)	SSD	0.5507	0.7641	0.6265
SDML	SSD	**0.6043**	**0.8179**	**0.6789**

Table 4: Comparison of different loss functions on Quora test set.

Loss	Dist	H@1	H@10	MRR
TL (Rand)	EUC	0.5738	0.7684	0.6428
TL (Rand)	SSD	0.6506	0.8579	0.7252
TL (Hard)	EUC	0.5549	0.7534	0.6256
TL (Hard)	SSD	0.5233	0.7077	0.5870
SDML	EUC	0.6526	**0.8832**	0.7361
SDML	SSD	**0.6745**	0.8817	**0.7491**

Table 5: Comparison of different loss functions on open domain QA dataset validation set.

Loss	Dist	H@1	H@10	MRR
TL (Rand)	EUC	0.5721	0.7695	0.6431
TL (Rand)	SSD	0.6538	0.8610	0.7271
TL (Hard)	EUC	0.5593	0.7593	0.6304
TL (Hard)	SSD	0.5201	0.7095	0.5863
SDML	EUC	0.6545	**0.8846**	0.7382
SDML	SSD	**0.6718**	0.8830	**0.7480**

Table 6: Comparison of different loss functions on open domain QA dataset test set.

gin. It is important to note that, since our dataset contains noisy examples, triplet loss with random sampling outperforms hard sampling setting, in contrast with the results presented in Manmatha et al. 2017.

The results presented in this section are consistent with our expectations based on the design of the loss function.

5 Conclusion

We investigated a variant of the paraphrase identification task - large scale question paraphrase retrieval, which is of particular importance in industrial question answering applications. We devised a weak supervision algorithm to generate training data from the logs of an existing high precision question-answering system and introduced a variant of the popular Quora dataset for this task. In order to solve this task efficiently, we developed a neural information retrieval system consisting of a convolutional neural encoder and a fast approximate nearest neighbour search index.

Triplet loss, a standard baseline for learning-to-rank setting, tends to overfit to noisy examples in training. To deal with this issue, we designed a new loss function inspired by label smoothing, which assigns a small constant probability to randomly paired question utterances in a training mini-batch resulting in a model that demonstrates superior performance. We believe that our batch-wise smoothed loss formulation will be applicable to a variety of metric learning and information retrieval problems for which triplet loss is currently widespread. The loss function framework we describe is also flexible enough to experiment with different priors - for e.g. allocating probability masses based on the distances between the points.

References

Delphine Bernhard and Iryna Gurevych. 2008. Answering learners' questions by retrieving question paraphrases from social q&a sites. In *Proceedings of the third workshop on innovative use of NLP for building educational applications*, pages 44–52. Association for Computational Linguistics.

Andrew P Bradley. 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*.

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 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jacob Goldberger, Geoffrey E Hinton, Sam T Roweis, and Ruslan R Salakhutdinov. 2005. Neighbourhood components analysis. In *Advances in neural information processing systems*, pages 513–520.

Iryna Haponchyk, Antonio Uva, Seunghak Yu, Olga Uryupina, and Alessandro Moschitti. 2018. Supervised clustering of questions into intents for dialog system applications. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2310–2321.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586.

Aminul Islam and Diana Inkpen. 2009. Semantic similarity of short texts. *Recent Advances in Natural Language Processing V*, 309:227–236.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, et al. 2016. Smart reply: Automated response suggestion for email. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 955–964. ACM.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Young-Bum Kim, Dongchan Kim, Anjishnu Kumar, and Ruhi Sarikaya. 2018. Efficient large-scale neural domain classification with personalized attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2214–2224.

Michael Levandowsky and David Winter. 1971. Distance between sets. *Nature*, 234(5323):34.

R Manmatha, Chao-Yuan Wu, Alexander J Smola, and Philipp Krähenbühl. 2017. Sampling matters in deep embedding learning. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2859–2867. IEEE.

Rada Mihalcea, Courtney Corley, Carlo Strapparava, et al. 2006. Corpus-based and knowledge-based measures of text semantic similarity.

Bhaskar Mitra and Nick Craswell. 2017. Neural models for information retrieval. *arXiv preprint arXiv:1705.01509*.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

Lutz Prechelt. 1998. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916. ACM.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. 2004. Simple bm25 extension to multiple weighted fields. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4144–4150. AAAI Press.

Kilian Weinberger, Anirban Dasgupta, Josh Attenberg, John Langford, and Alex Smola. 2009. Feature hashing for large scale multitask learning. *arXiv preprint arXiv:0902.2206*.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.