Neural Computing and Applications
https://doi.org/10.1007/s00521-020-05530-1

S.I. : EFFECTIVE AND EFFICIENT DEEP LEARNING

Toward hardware-aware deep-learning-based dialogue systems

Vlad Pandelea[1] · Edoardo Ragusa[2] · Tom Young[1] · Paolo Gastaldo[2] · Erik Cambria[1]

Received: 8 June 2020 / Accepted: 10 November 2020

Abstract
In the past few years, the use of transformer-based models has experienced increasing popularity as new state-of-the-art performance was achieved in several natural language processing tasks. As these models are often extremely large, however, their use for applications within embedded devices may not be feasible. In this work, we look at one such specific application, retrieval-based dialogue systems, that poses additional difficulties when deployed in environments characterized by limited resources. Research on building dialogue systems able to engage in natural sounding conversation with humans has attracted increasing attention in recent years. This has led to the rise of commercial conversational agents, such as Google Home, Alexa and Siri situated on embedded devices, that enable users to interface with a wide range of underlying functionalities in a natural and seamless manner. In part due to memory and computational power constraints, these agents necessitate frequent communication with a server in order to process the users' queries. This communication may act as a bottleneck, resulting in delays as well as in the halt of the system should the network connection be lost or unavailable. We propose a new framework for hardware-aware retrieval-based dialogue systems based on the Dual-Encoder architecture, coupled with a clustering method to group candidates pertaining to a same conversation, that reduces storage capacity and computational power requirements.

Keywords Dialogue systems · Natural language processing · Artificial intelligence

1 Introduction

Recently, the release of deep pre-trained transformer [1] based models [2, 3] has led to improved performance in fields such as natural language processing [4], knowledge representation [5], commonsense reasoning [6], personality detection [7] and more. The typical approach sees these large models used as the base on top of which simple architectures, such as a classification layer, specialized for the task at hand are placed. The added architecture is then trained for the specific task, and the deep transformer base is fine-tuned at the same time. In most cases, however, these kind of models are not suited to be placed on embedded devices where resources are limited. In fact, deep learning solutions require custom hardware [8], or aggressive pruning and quantization strategies that affect generalization performance [9]. These limitations can be expressed in terms of both memory and inference speed [10]. Moreover, relying on communication with a server where larger models can be executed, and then the result fetched, leads to additional concerns in terms of network availability and privacy [11, 12].

In this paper, we address some of the aforementioned issues in the context of dialogue systems, a domain that has recently attracted increasing attention thanks to new developments in AI research. In contrast with previous architectures characterized by a clear-cut separation of

✉ Erik Cambria
cambria@ntu.edu.sg

Vlad Pandelea
vlad.pandelea@ntu.edu.sg

Edoardo Ragusa
edoardo.ragusa@edu.unige.it

Tom Young
tom@sentic.net

Paolo Gastaldo
paolo.gastaldo@unige.it

[1] School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore

[2] Department of Electrical, Electronic, Telecommunications Engineering and Naval Architecture, DITEN, University of Genoa, Genoa, Italy

Published online: 07 January 2021

different components [13], the rapid increase in publicly available conversational data and the availability of powerful deep learning techniques have recently spurred the study of holistic data-driven approaches to push the boundaries of chatbots on different fronts including task-oriented dialogue systems [14], knowledge-augmented dialogue systems [15], multimodal dialogue systems [16], empathetic dialogue systems [17], and conversational sentiment analysis [18].

In particular, retrieval-based dialogue systems [19–21] work by finding the best candidate among a set of predefined candidate responses, given a context. While being limited in the set of possible responses, they are able to provide syntactically rich and more diverse responses than their generative counterpart. At the same time, this poses several challenges. For example, in order to identify the best candidate, in principle all of the possible candidates must be evaluated with respect to the current context, which may result in prohibitive inference time. When this kind of system is to be deployed on embedded resources-constrained devices, additional challenges arise. Firstly, the computational power is further reduced. Secondly, it might not be practical to maintain a list of all candidates on the local storage, due to memory constraints. The solution that we propose in this paper splits the candidate search in two phases. During the training phase, the candidate responses are divided in a set of clusters. During the inference phase, a cluster identifier (CI) selects a subset of the candidate responses by implementing a classification function that assigns the novel context to a specific cluster. Notably, cluster-based distribution is particularly suitable for devices with multiple level of memories characterized by different speed/dimension trade-off. The retrieval model selects the best response looking only at the selected subset of responses, i.e., the one belonging to the selected cluster.

The clustering procedure aggregates data into clusters by using a similarity measure induced by a binary classifier model with the task of determining whether two sentences belong to a same dialogue. This metric is defined in order to achieve two desirable properties. Firstly, the inference process is accelerated as the cluster metric definition hinges on contexts and responses belonging to a same cluster. Secondly, the clusters defined by this metric are also characterized by wider sense of locality beyond the individual context and response, in the sense that we are likely to find sentences within a same dialogue, even at a greater distance, within a same cluster. These two properties are well-suited for a system that is to be placed on embedded devices, as they address two of their major limitations, namely computational power and storage capacity.

1.1 Contribution

The work presented in this paper defines a new framework for hardware-aware retrieval-based dialogue systems. The main contributions can be summarized as follows:

- The definition of a hardware-aware retrieval-based dialogue system framework. The proposed framework allows a partitioning of the candidate responses that allows faster inference and enables hierarchical memory solutions. The first property, faster inference, is desirable for embedded devices as these portray a scenario with reduced computational power. Moreover, it can help to lengthen battery life with respect to a traditional system. The second property is desirable as storage capacity is often a limitation, in embedded devices, that calls for ad-hoc solutions such as the one that we propose in this paper. Note that since the method that we propose makes no strong assumptions on the underlying hardware, it is suitable in all those cases in which multiple levels of memory are available in any form (e.g., on a device itself, or via communication with a server). Moreover, even within a single memory, the method may still be used to accelerate the inference process.
- A clustering strategy for the candidate responses based on a data driven similarity measure. Importantly, the similarity measure is learned directly from the training data, optimizing the criterion that we set, based on the properties that we wish the final clustering to possess. In our case, since the overall framework is tailored to the requirements of embedded devices, we choose a criterion that promotes a higher similarity for sentences that belong to the same conversation, in order to minimize the need for memory updates.

Additionally, we show that:

- Small-margin knowledge distillation [22] from a pre-trained network continues to be effective in the dialogue retrieval task, and can lead to reliable improvement at a negligible cost.
- The recently released Poly-Encoder model [23] continues to outperform the original Dual-Encoders [19] by a significant margin in most cases and on different tasks using a GRU base instead of a transformer, while maintaining the possibility to pre-compute candidate embeddings.
- A naïve form of multitask learning [24] to jointly learn the dialogue retrieval function and the clusters similarity metric, despite performing worse on the individual tasks, leads to the number of parameters to be roughly halved, which can prove useful in the case of user-specific training of the system to better tailor the model to individual needs based on user profile.

2 Related work

2.1 Retrieval-based dialogue systems

Retrieval-based dialogue systems can be placed under the umbrella term of candidate scoring models, where the most suitable candidate, given an input context, is to be found. One of the ways in which these models can be categorized is by the type of interaction that context and candidates undergo. Specifically, in the case of Dual-Encoders [19, 25], context and candidate are encoded through separate networks and then combined, e.g., by means of a dot product, to obtain the final output.

In the case of Cross-Encoders [26, 27], context and candidate are combined before the encoding takes place, permitting to model more complex interactions. At the same time, Cross-Encoders fall short when it comes to inference speed, as by requiring both context and candidate to perform the forward pass, pre-computing the encodings is not possible. Recently, in [23], the Poly-Encoder architecture that we also use in this work was introduced. Poly-Encoders attempt to maintain the speed of Dual-Encoders by encoding the candidate responses in the same fashion, whereas the context encoding also utilizes information from the response candidate, thus allowing for a more rich set of interactions than Dual-Encoders. In particular, in order for the Poly-Encoder architecture to capture the advantages of both the Dual-Encoder and the Cross-Encoder architecture, it does the following: to achieve inference speed comparable to the Dual-Encoder, it maintains the possibility to pre-compute the candidate response embeddings by using a standalone architecture block that does not receive inward information from the context embedder block. To model more complex interactions, in a similar fashion to the Cross-Encoder, it uses the candidate response embeddings to attend over the context.

2.2 Top candidate selection

The importance of speeding up the inference process for top candidate selection, and softmax-based classifiers in particular, was previously acknowledged. In [28, 29] the authors proposed to reduce Maximum Inner Product Search (MIPS) to nearest neighbor search (NNS) and then solve NNS by Locality Sensitive Hashing (LSH). A database partitioning scheme was introduced in [30]; however, the effectiveness of the method becomes low for high-dimensional data. Graph-based solution was also proposed [31–33]. In these solutions, the candidate responses are reorganized in a graph and top responses are selected based on graph searching procedures. Finally, solutions targeting the approximation of softmax, such as MIPS, have been presented [34, 35] explored quantization to limit inference time; meanwhile, [36] proposed a greedy solution.

The speedup of top candidate selection by means of clustering of the context vectors was proposed in [37]. More specifically, the authors proposed a screening model that reduces the number of candidate responses to consider, while preserving high accuracy. The method uses a k-means clustering algorithm trained on the similarity measure induced by the softmax classifier. In this work, the retrieval model is optimized on pairwise similarities, i.e., the cost function penalizes pairwise errors; meanwhile, softmax considers similarity of all patterns simultaneously. As a consequence, the induced space is less likely to cluster group of similar data close to each other in a globular manner. Remarkably, spectral clustering works directly on the similarity matrix. As a consequence, it is expected to identify the best clusters for the pairwise metric defined by the binary classification model, i.e., the metric that measures the likelihood that two sentences belong to the same conversation.

3 Approach

The paramount goal of this paper is the development of hardware-aware retrieval-based dialogue systems. In order to achieve this goal, the set of candidate responses is divided in subsets, based on a clustering procedure. During the inference phase, a filtering model named Cluster Identifier (CI) selects, based on the incoming patterns, a cluster that is to be further explored in looking for the appropriate response. Thus, the retrieval model selects the best response by checking only a subset of the overall set, the subset contained in the cluster that is selected by the CI. This solution has two important properties. Firstly, the number of comparisons to be performed in identifying the best candidate is reduced, reducing inference time and power consumption. Secondly, this partitioning of data in clusters enables the use of multiple levels of memory. By doing so, the complete set of data can be hosted in a large and slow memory. At the beginning of the conversation, the cluster selected by the CI can be loaded in a smaller and faster memory. The initial overload in time due to the fetch and load of the cluster in the first phase will be balanced later with the better performance of the smaller memory. To achieve these two properties, we build the clusters following the method explained in Sect. 3.2.

Figure 1 summarizes the overall inference system. The CI retrieves the selected subset from the database. Then, the retrieval model selects the best response among the small chunk selected. The figure highlights the logical division among the database and the Selected Subset that can be mapped to two different storage solutions. As an example, in a smartphone, the Database could be hosted in

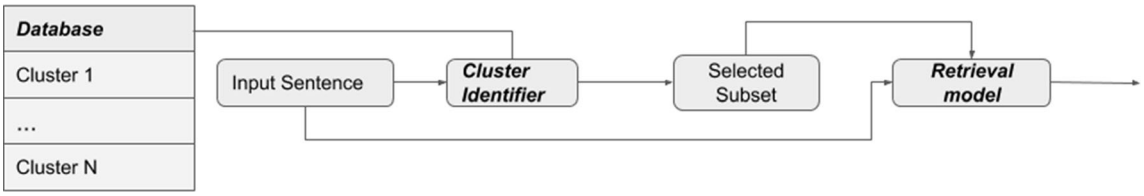

Fig. 1 Overall architecture of the proposed system

a ssd memory; meanwhile, the selected subset could be hosted in a small portion of the RAM memory.

The following two subsections describe in detail the retrieval model and the clustering strategy employed to implement the overall system.

3.1 Dialogue retrieval model

The Dialogue Retrieval Model (Retrieval Model in Fig. 1) selects the response of the algorithm by searching the best match for the input context among a set of candidates, by computing the pairwise similarity between the input context and each response candidate.

As our main retrieval model, we choose to employ the Dual-Encoder architecture with a couple of differences with respect to the one originally proposed in [23]. Firstly, as the base encoder we do not use BERT, but we instead use a Gated Recurrent Unit (GRU) [38] base as to keep the model lightweight. Secondly, we use the last two hidden states of the GRU encoder still results in a computational cost lower than the Poly-Encoder while showing better performance in our experiments.

Specifically, the context $c = c_1, .., c_{nc}$ and the candidate response $r = r_1, .., r_{nr}$ are first encoded into $h^c = h^c_1, .., h^c_{nc}$ and $h^r = h^r_1, .., h^r_{nr}$, respectively:

$$h^c_{nc} = \mathbf{GRU_c}(h^c_{nc-1}, \mathbf{e}(c_{nc})) \quad (1)$$

$$h^r_{nr} = \mathbf{GRU_r}(h^r_{nr-1}, \mathbf{e}(r_{nr})) \quad (2)$$

where $\mathbf{e}(x)$ denotes the GloVe [39] embedding of word x. The score is determined as $p_r = [h^c_{nc-1}; h^c_{nc}]^T [h^r_{nr-1}; h^r_{nr}]$, and the cost function used is the Binary Cross Entropy (**BCE**) between the prediction and the ground truth.

3.2 Cluster identifier

The CI identifies the clusters that are more likely to contain appropriate responses. The process is divided in two stages. Offline, firstly a similarity measure is optimized on the training data. Then, the set of responses is divided into clusters and stored in a database. Online, a CI selects the interesting clusters based on the current context.

The quality of the clustering of the response candidates plays a key role in the proposed system. In principle, in order to have an effective solution the clustering algorithm should group all the sentences belonging to a same conversation into a same cluster. The major point to obtain an effective clustering is the definition of a good notion of similarity. The proposed similarity measure is obtained by optimizing an encoder structure on the training data. In addition, the clustering strategy should trade-off the clusters' dimension and database coverage. On the one hand, a small cluster implies a small number of comparisons for the Retrieval Model, leading to fast inference. On the other hand, if the cluster is too small, it is possible that the original dataset is not sufficiently represented. Accordingly, the solution would select a suboptimal response. Also note that heuristics based on threshold values for the confidence of the Retrieval Model can be easily applied to expand the search when the selected subset is deemed not representative.

In this work, the encoder shares the same architecture of the retrieval model described in Sect. 3.1, but the training strategy is slightly different. More specifically, the encoder addresses the following binary classification problem. It receives two input sentences, and is trained with the objective of predicting 1 if the sentences belong to the same conversation and 0 otherwise. Once the encoder is trained, the similarity matrix containing the pairwise similarity of the data is computed. The size of this matrix grows quadratically with the number of sentences that are to be clustered. With the increasing of the size of the training set, the number of operations rapidly becomes too big. Therefore, we use a sampling strategy so that each sentence is compared only with a small subset of the training data. In particular, from each conversation, we extract a subset of k sentences. The selected subset is chosen as the subset of k sentences that obtained the average highest score when compared with the sentences of the same conversation using the binary classification metric. The underlying rationale is to filter out non-informative sentences and reduce the computational cost.

Previous approaches explored the use of k-means clustering using the hidden representation of the input data inside the encoder. However, it should be noted that (1) the encoder is trained on a subset of the training data and (2) the loss function measures only pairwise similarities. As a consequence, there are no guarantees about the overall

distribution of the sentences in the remapped space. Spectral clustering instead works directly on the similarity matrix. The key point is that it works on pairwise similarities; as a consequence, it is expected to better fit the space induced by the proposed metric.

Assigning a novel datum to a cluster is not straightforward because clusters do not have explicit centroids as in the case of k-means [40]. To overcome this issue, a classifier is implemented. Given a training set $\boldsymbol{\mathcal{X}}$, a label set $\boldsymbol{Y}$ is built based on the clustering results. Then, the eventual classifier is trained on the classification problem ($\boldsymbol{\mathcal{X}}$, $\boldsymbol{Y}$).

The CI classification system employs a **LSTM** and a fully connected layer followed by a softmax:

$$p_c = \textbf{softmax}(Wh + b) \qquad h = \textbf{LSTM}(\mathbf{e}(c)) \tag{3}$$

where c is the context. The clustering mechanism is detailed in Algorithm 1.

trained teacher network to instill additional task specific knowledge into the smaller model by promoting an output level behavior where the latter model mimics the behavior of the former. Since the knowledge is instilled by only observing the output of the larger model, this method is architecture-agnostic [41]. To accomplish this, we use a pre-trained Cross-Encoder to obtain the additional score labels $s_1^e, .., s_n^e$ where n is the number of elements in the training set. The complete cost function for a (*context*, *response*) pair (c_i, r_i) is briefly defined as follows:

$$\mathbf{L}(c_i, r_i) = (1 - \lambda)\textbf{BCE}(p_{r_i}, s_{r_i}) + \lambda\textbf{MAE}(p_{r_i}, s_{r_i}^e) \tag{4}$$

where p_{r_i} is the score given to candidate response r_i by the retrieval model, λ is a trade-off parameter, **BCE** denotes Binary Cross-Entropy with logits, **MAE** the Mean Absolute Error, and s_{r_i} is 1 if r_i is the response to c_i and 0 otherwise.

Algorithm 1 Clusters Construction Mechanism

Input A training corpus $\mathcal{R}$ containing N sentences divided in C conversations;

1. **Similarity Notion Training**
(a) Extract N_{true} true pairs $\{c_i, r_i\} \in$ same conversation
(b) Extract N_{false} false pairs $\{c_i, r_i\}$ sampling from different conversations
(c) Train the similarity notion $f(s_1, s_2)$. The similarity notion is trained on the binary task: s_1, $s_2 \in$ same conversation. The training set is composed of the $N_{true}+ N_{false}$ pairs previously extracted.

2. **CI Training**
(a) Compute similarity matrix using a subset of $\tilde{N}$ sentences: $S \in R^{\tilde{N}\times\tilde{N}}$
$s_{i,j} = [f(s_i, s_j) + f(s_j, s_i)]/2$;
(b) Execute spectral clustering on the similarity matrix S obtaining the labeled training set $\{\tilde{\mathcal{R}}, \tilde{Y}\}$, where Y is the label vector denoting the cluster of each sentence.
(c) Train the CI using the labeled training set $\{\tilde{\mathcal{R}}, \tilde{Y}\}$

3. **Clusters Construction**
For each novel candidate response $r_i, .., r_m$:
(a) Obtain the most probable cluster $k = \textbf{CI}(r_i)$
(b) Assign r_i to cluster k

Output Set of labeled candidate responses $\{(r_1, k_1), .., (r_m, k_m)\}$ where k_i is the cluster that response r_i has been assigned to in 3.

3.3 Training optimization

To boost the performance of our model, we experiment with two approaches that present a cost only during the initial training phase. Firstly, we experiment with a knowledge distillation approach [41] whereby the cost function that is minimized during the training phase also takes into account the labels predicted by a larger and more accurate model. The main idea behind this approach is that of leveraging the pre-

Secondly, to attempt to further reduce the number of parameters, which might have significant impact in the case of user-specific model training, we experiment with a simple multitask learning approach whereby we employ the same model for the retrieval task and for the clustering similarity metric definition. In particular, we alternate between the two tasks that are discussed in Sects. 3.1 and 3.2 (Binary Classification) at every batch during the training phase. We experiment with two settings. In the

first setting, we naïvely use a shared final layer between the two tasks that alternate at each batch. In the second setting, we place task-specific projection layers, that re-map the final context representation and the response representation into a new vector space, of the same dimension, based on the task. In particular, the context representation in Eq. 1 and the response one in Eq. 2 undergo a further transformation based on the task, as follows:

$$\begin{aligned} h^c_{t_1} &= \tanh(W_{p_1,t_1}[h^c_{nc-1}; h^c_{nc}]) \\ h^r_{t_1} &= \tanh(W_{p_2,t_1}[h^r_{nr-1}; h^r_{nr}]) \end{aligned} \tag{5}$$

$$\begin{aligned} h^c_{t_2} &= \tanh(W_{p_1,t_2}[h^c_{nc-1}; h^c_{nc}]) \\ h^r_{t_2} &= \tanh(W_{p_2,t_2}[h^r_{nr-1}; h^r_{nr}]) \end{aligned} \tag{6}$$

where $W_{p_1,t_1}, W_{p_2,t_2}, W_{p_1,t_2}, W_{p_2,t_2}$ are model parameters and tanh the hyperbolic tangent. The score is determined as $p_{r,t_1} = (h^c_{t_1})^T h^r_{t_1}$ and $p_{r,t_2} = (h^c_{t_2})^T h^r_{t_2}$ for the first and second task, respectively.

3.4 Computational cost analysis

In this section, the computational cost of the proposed solution is analyzed. Without loss in generality, we first consider the case where a single memory supports all the computations. Later, we analyze a solution with multiple levels of memory, pointing out the trade-off between speed and memory consumption.

Suppose that a set of N responses is divided in K clusters, where for each response the corresponding clusters label is specified $\{\boldsymbol{\mathcal{R}}, \boldsymbol{Y}_{\boldsymbol{\mathcal{R}}}\}$. Let N_k be the number of responses contained in the k-th cluster.

Using the standard implementation, the eventual classifier requires the computation of the score, via the retrieval model, for all the N candidate responses. In practice, the computational cost grows as:

$$\boldsymbol{O}_{\text{base}} = N \times \boldsymbol{O}_{\text{RM}} \tag{7}$$

where $\boldsymbol{O}_{\text{RM}}$ is the computational cost of a single inference phase of the retrieval model.

The proposed algorithm splits the inference phase in two parts. Firstly, the CI ranks the clusters based on the likelihood that a cluster contains the correct response. Later, the retrieval model scores the context against the subset of candidate responses identified by the CI. As a consequence, the computational cost becomes:

$$\boldsymbol{O}_{\text{proposal}} = \boldsymbol{O}_{\text{CI}} + \sum_{k=1}^{N_S} N_k \boldsymbol{O}_{\text{RM}} \tag{8}$$

where N_s is the number of clusters that are involved in the retrieval operation. For example if $N_s = 2$ the retrieval model scores the context against the responses belonging to the two clusters with maximum likelihood only.

Given that $\boldsymbol{O}_{\text{CI}} \simeq \boldsymbol{O}_{\text{RM}}$, the speedup of the proposed solution is:

$$G = \frac{\boldsymbol{O}_{\text{base}}}{\boldsymbol{O}_{\text{proposal}}} = \frac{N}{1 + \sum_{k=1}^{N_S} N_k} \tag{9}$$

Equation 9 points out that the eventual speedup is proportional to the subset of clusters that are involved in the retrieval phase. The eventual speedup is maximized when cluster sizes are uniformly distributed $N_k = N/K$. Experimental results shown in the following demonstrate that good performance in terms of accuracy and recall can be obtained for small values of N_s.

From a memory point of view the proposed solution stores almost double of parameters for models, again assuming that the number of parameters of the CI model $\boldsymbol{M}_{\text{CI}}$ is about the same as $\boldsymbol{M}_{\text{RM}}$, i.e., the number of parameters of the retrieval model. However, the major memory contribution is given by the set of responses.

Retrieval-based dialogue systems are memory hungry by definition. In most of cases, commercial embedded devices do not embed enough RAM memory to host the entire set of responses. As a consequence, only a small portion of the data can be loaded. Loading data from external memories, or even servers, is a time demanding operation and rapidly becomes the computational bottleneck in most of the applications if frequent updates are required.

The present solution clusters data based on the probability that two sentences belong to the same conversation. In an ideal configuration, the CI should select one cluster at the beginning of the interaction and maintain the same configuration throughout the conversation. In doing so, data fetching would be minimized, rendering the proposed solution particularly suitable for memory constrained devices. In practice, relying solely on the initial message to select the clusters that will be used throughout the conversation is unlikely to be an optimal strategy. As the conversation unfolds, topics are likely to shift, therefore a more realistic approach would see the clusters being periodically updated based on heuristics such as the last time they have been used, or other strategies inspired to existing cache update mechanisms. Lastly, note that the method proposed in this paper scales consistently with dataset size and that the number of clusters depends on the variety of conversational patterns rather than on their number.

4 Experiments

4.1 Dataset

To test our system, we choose to employ the two datasets that we now briefly describe. Firstly, DailyDialog [42] is a textual dataset consisting of day-to-day conversations. The

conversations in this dataset are more similar to real life daily conversation, as opposed to datasets such as those based on Twitter [43], as they are crawled from websites where English learners practice day-to-day dialogue. In addition, each conversation focuses on a specific topic, as in real life situations. In all of our experiments we use the data splits provided, adapting them as needed for the task. In total, this dataset contains 13118 dialogues and an average of 7.9 speaker turns. Each turn contains an average of 14.6 tokens. Secondly, the Ubuntu Dialogue Corpus [19] is one of the largest and most widely used conversational datasets. This corpus consists of around one million dyadic conversations concerning technical support for various Ubuntu-related problems. We use the original split containing 1 million training instances. We use this dataset to validate our dialogue retrieval component. For all other experiments, we use the DailyDialog corpus.

4.2 Experimental details

4.2.1 Retrieval task

The DailyDialog dataset format is not suited to be used for a dialogue retrieval task. Therefore, we adapt it by creating pairs of context and candidate responses for the training set, where the label is 1 if the candidate response is the utterance that follows the context, and 0 otherwise. Similarly to [44], we use an uneven ratio of positive to negative samples. In particular, we use four randomly sampled responses to be labeled as 0 by the model for every correct response to be labeled as 1. The negative samples are chosen by sampling uniformly from the training set, with the exclusion of the correct response. For the validation set, in a similar fashion, we sample 9 negative samples for every positive one in order to evaluate the model based on recall@k, that is the percentage of times that the correct response is ranked as relevant by the model among a total of k possibilities.

For the Ubuntu Dialogue Corpus, we use the original split provided, with the difference that we choose to employ word embeddings of dimension 100, as opposed to the 300 dimensional ones in the original work.

For DailyDialog we use a maximum context length of 128 tokens, and a maximum candidate response length of 32, using a leading padding strategy. For Ubuntu, we use 160 tokens for the context and 32 for the response.

The model is trained with the Adam optimizer [45], a learning rate of 0.0009, batch size 16 and a maximum of 6 epochs. m, the number of randomly initialized codes, is set to 16. The best model is chosen at the point where recall 1@10 is maximized on the validation set.

For this task, we experiment with three models, all of which allow the pre-computation of response candidates, which is fundamental for fast inference. First is the dual-encoder, which processes context and response independently. Second is the original Poly-Encoder model, and third is our variation of the Dual-Encoder.

4.2.2 Binary classification task

For the binary classification task, that is at the core of the similarity matrix used in building the clusters, we use the same Dual-Encoder-based model used for the retrieval task. The input this time consists of context, candidate response pairs where the label is 1 if the context and the candidate response belong to the same conversation, and 0 otherwise. We randomly sample a maximum of three instances where the label is positive and three where it is negative for each conversation.

In this case, we use a maximum length of 32 tokens for both context and candidate response. Again, we use the Adam optimizer with learning rate set to 0.0009 and batch size 16. The maximum number of epochs is 8, and the best model is chosen when the accuracy is maximized on the validation set.

As for the retrieval task, here we use the same three models, as to allow further experimentation with multitask learning between the two tasks.

4.2.3 Multitask learning

For the multitask learning setting, we use the configuration that performs best on the retrieval task. Same batch size and learning rate is used for both tasks, 16 and 0.0009, respectively. Maximum number of epochs is set to 8, and all of the other settings are as in the individual tasks. At every step we train on a batch of each of the tasks, and validate on each of the tasks separately, keeping track of the best performance on each.

4.2.4 Clustering task

The selected subset of the training data is clustered using various number of clusters in order to analyze the trade-off between computational cost and performance. The clustering process is performed using the spectral clustering algorithm directly on the similarity matrix obtained comparing the training data using the similarity metric induced by the binary classification task. Computations are constrained to `fp32` data format to reduce memory consumption.

The CI is trained on the labeled dataset obtained by the clustering procedure. Training parameters are set to 100 hidden units for the LSTM, 32 as batch size, a learning rate of 0.0009, a maximum of 5 epochs and Adam as the optimizer.

K-Means-based baseline In addition to our main method, we also propose a K-Means-based baseline.

Specifically, we average the output of the two **GRU** encoders of the binary classifier, that builds the similarity matrix, to obtain the representation for each utterance. We then apply the K-Means algorithm to obtain the clusters.

4.3 Results

In this section, we first analyze the performance of the individual components that constitute our complete system and then evaluate the overall system itself.

4.3.1 Dialogue retrieval

To evaluate the Dialogue Retrieval component of our system, we use the recall@k metric. We experiment with various configurations varying the number of hidden units in the **GRU** layer. Complete results are shown in Table 1. In the first column, we report the model names, in the second one the metrics, in the first row the dataset, and in the second one the number of hidden units used.

We notice that despite performing slightly worse on the Ubuntu corpus, both the Poly-Encoder and the modified Dual-Encoder perform significantly better than a Dual-Encoder baseline on DailyDialog. Moreover, the modified Dual-Encoder surpasses the recall of the Poly-Encoder in all cases.

Knowledge Distillation We pick the best model, our Dual-Encoder with 150 units, and explore the impact of knowledge distillation from a simple BERT-based Cross-Encoder, following [23], that scores 0.77 on Recall 1@10. In particular, we experiment with different values of the λ parameter (Eq. 4) that decides the trade-off between importance given to the original label and to the soft one obtained from the Cross-Encoder. The results of this experiment are shown in Fig. 2. We find that a moderate λ between 0.35 and 0.5 leads to consistent improvement,

Table 1 Recall 1@10, 1@2 and 5@10

	Units	DailyDialog			Ubuntu
		40	110	150	150
Dual-Encoder (ours)	R1@10	0.61607	0.67746	0.70095	0.62821
	R1@2	0.9031	0.92304	0.92545	–
	R5@10	0.95714	0.96902	0.97128	–
Poly-Encoder	R1@10	0.59301	0.66997	0.68708	0.62796
	R1@2	0.89277	0.9205	0.92333	–
	R5@10	0.94964	0.96704	0.97015	–
Dual-Encoder	R1@10	0.59655	0.65427	0.66091	0.63499
	R1@2	0.89673	0.9164	0.91442	–
	R5@10	0.95445	0.96195	0.95869	–

Underlined the best performance for each metric

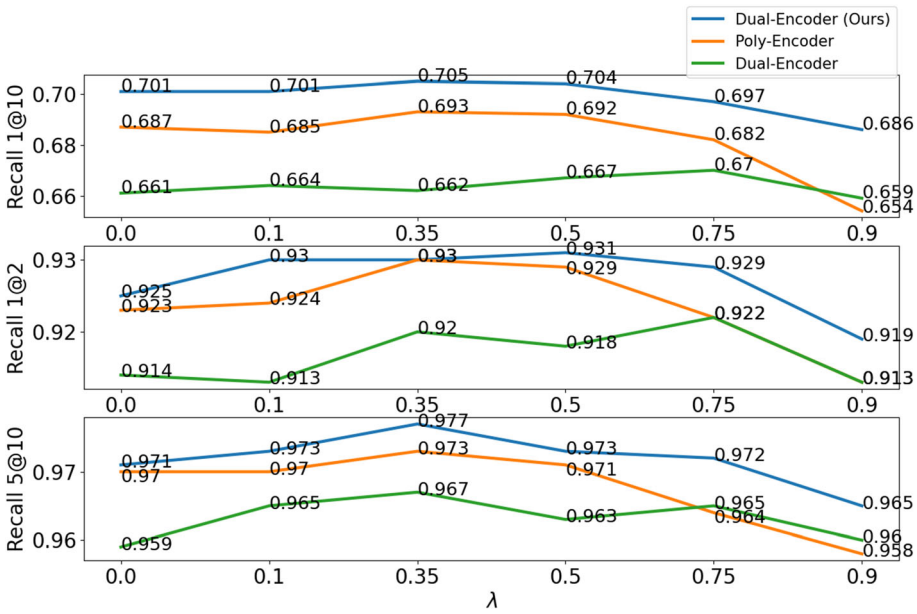

Fig. 2 λ denotes the weight given to the soft labels as per the cost function in Eq. 4

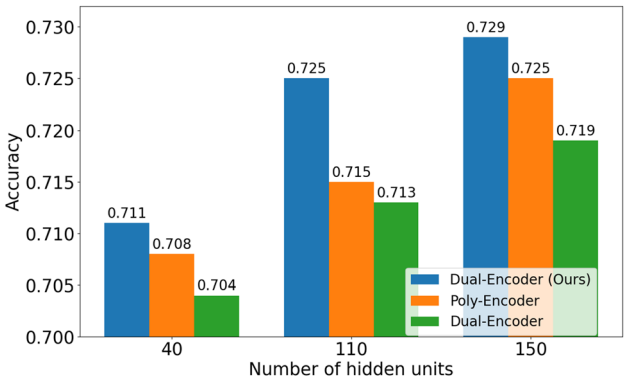

Fig. 3 Binary classifier performance

whereas a higher one often leads to degradation of the performance, possibly due to the relatively small margin between the two classifiers.

4.3.2 Clustering

Similarity matrix definition via Binary Classifier We report the results, in terms of accuracy, of the Binary Classifier for various dimensions of the **GRU** layer hidden size in Fig. 3. On the *x*-axis we show the number of units for different configurations of the various models, whereas on the *y*-axis we show the accuracy. Similarly to the retrieval task, we find that our Dual-Encoder has the best performance, and both this variation of the Dual-Encoder and the Poly-Encoder outperform the original Dual-Encoder.

Multitask Learning In Table 2, we show the results of the two setting using our Dual-Encoder, with and without the projection layer. We notice that the performance on the individual tasks drops, and that using a projection layers leads to further degradation of both recall 1@10 and accuracy. This approach leads to a loss of performance, however having a joint model halves the number of parameters required to be tuned.

Clustering In Table 3, we report the results for the clusters creation task, as percentage of dialogues that have at least 50% of the sentences in the same cluster, as the number of training patterns varies. In the table, the best results on this metrics are shown; however, in the subsequent experiments we take the models that perform best with an additional constraint in terms of distribution of elements within the clusters. In particular, we constrain the maximum share of elements in a single cluster. In Table 4, we report the same metric reported in Table 3 for the models that achieve the best result while satisfying the constraint specified in the *Max Share* column. In Table 5, we report a thorough analysis for the models of which in Table 4.

We notice that the clusters created through the K-Means baseline (**KM**) tend to have elements that are more evenly distributed, in terms of the standard deviation of the clusters distribution (*Std*). However, when we look at *Dist*, which denotes the average distance between the cluster that the correct response resides in, and the one the context is assigned to, we find that the CI-based method vastly outperforms the **KM** alternative.

Acc denotes the accuracy of the model in picking the cluster in which the response to the context is contained. Again, we find that the CI method performs significantly better and more consistently as the number of clusters increases. *R@10C* and *R@10A* denote, respectively, the recall 1@10 when the 9 negative samples are chosen from

Table 3 At most N patterns from each train dialogue: numbers represent the percentage of validation dialogues with at least 50% of the sentences in the same cluster

	3	4	5	6
Dual-Encoder (ours) - 5	0.86887	0.85085	0.84184	0.86987
Dual-Encoder (ours) - 10	0.71772	0.78478	0.72673	0.73674
Dual-Encoder (ours) - 20	0.6006	0.63463	0.61461	0.5996
Dual-Encoder (ours) - 50	0.4344	0.5465	0.52452	0.4924

Table 4 Percentage of validation dialogues with at least 50% of the sentences in the same cluster, constrained by the most populated cluster containing less than the indicated percentage of elements

	Score	Max share (%)
Dual-Encoder (ours) - 5	0.8699	40
Dual-Encoder (ours) - 10	0.7367	30
Dual-Encoder (ours) - 20	0.6146	20
Dual-Encoder (ours) - 50	0.4545	10

Table 2 Multitask approach using Dual-Encoder with 150 hidden units

	Shared #1	Shared #2	Projection #1	Projection #2
Retrieval	0.65851	0.64946	0.59202	0.58452
Binary classifier	0.7015	0.7135	0.6835	0.6951

For the retrieval task we report the recall 1@10, whereas for the Binary Classifier we report the accuracy. Shared indicates that the final layer is shared, projection that we have an additional layer, based on the task, before the final inner product. #1 denotes the results when the retrieval performance is maximized, #2 the results when the binary classification performance is maximized

Table 5 Statistics on the final system

	CI-5	CI-10	CI-20	CI-50	KM-5	KM-10	KM-20	KM-50
Std	0.1072	0.0663	0.0332	0.0177	0.0912	0.0523	0.0134	0.0072
Dist	0.9164	1.7403	3.6925	9.3062	1.4103	3.7717	8.4035	23.5118
Acc	0.5341	0.4557	0.3612	0.2773	0.3965	0.2020	0.1223	0.0423
R@10C	0.6062	0.5766	0.532	0.484	0.67	0.6645	0.6433	0.6687
R@10A	0.7418	0.7607	0.8009	0.8230	0.7511	0.7481	0.7495	0.7914
R@C	0.0374	0.0366	0.0374	0.0351	0.0343	0.0236	0.0210	0.0138
R@A	0.0374	0.0374	0.0374	0.0374	0.0374	0.0374	0.0374	0.0374

CI our approach comprising the CI, **KM** the K-Means-based approach

the correct cluster, and when they are chosen randomly. We evaluate this only when the correct cluster is chosen by the CI/**KM**. In all cases, *R@10C* is lower than *R@10A*, suggesting that responses within the clusters are more related, hence the drop in performance of the retrieval model. This phenomenon is more accentuated for the CI strategy, indicating a better clustering outcome in terms of elements relatedness. Additionally, we notice that *R@10A* is significantly higher than the results shown in Fig. 2, perhaps indicating that contexts for which it is easier to identify the correct cluster correspond to responses that are easier to identify for the retrieval model as well, possibly due to a more substantial information content. *R@C* and *R@A* indicate the recall when we look at the whole set of possible candidate responses (3851), in the first case using our strategy, and in the second by simply comparing all of the candidate responses. We notice that our strategy maintains the recall almost unaltered, whereas the **KM** mechanism sacrifices increasingly more performance in terms of recall as the number of clusters increases.

Next, we analyze the speedup that the various configurations yield, as average number of comparisons to be performed, over a baseline with no clusters, or equivalently, one cluster containing all of the candidate responses. In the left plot of Fig. 4 we show the speedup comparison between the two methods as the number of clusters varies. We find that both proposed strategies lead to a significant speedup, up to 44.2 times in the case of **KM** and 27.98 in the case of CI, with respect to the naïve strategy of comparing a context with all candidate responses individually.

Lastly we investigate the degree of locality, as the average number of contexts per cluster for each conversation. The lower this number, the more clusters we need to look in during a conversation. This is shown in the right plot of Fig. 4. We find a significant difference between the CI and **KM** approach, showing a much higher degree of within-conversation locality for the former.

5 Conclusion

In this work, we proposed a new framework for hardware-aware retrieval-based dialogue systems that aims to provide increased inference speed and reduce memory requirements during a conversation. Our system achieves these two proprieties by devising a clustering method based on a similarity metric induced by pairwise distances as defined by a specifically trained neural model set to promote lower distances for utterances pertaining to a same dialogue. Experimental results show that the use of the proposed method maintains good performance in widely

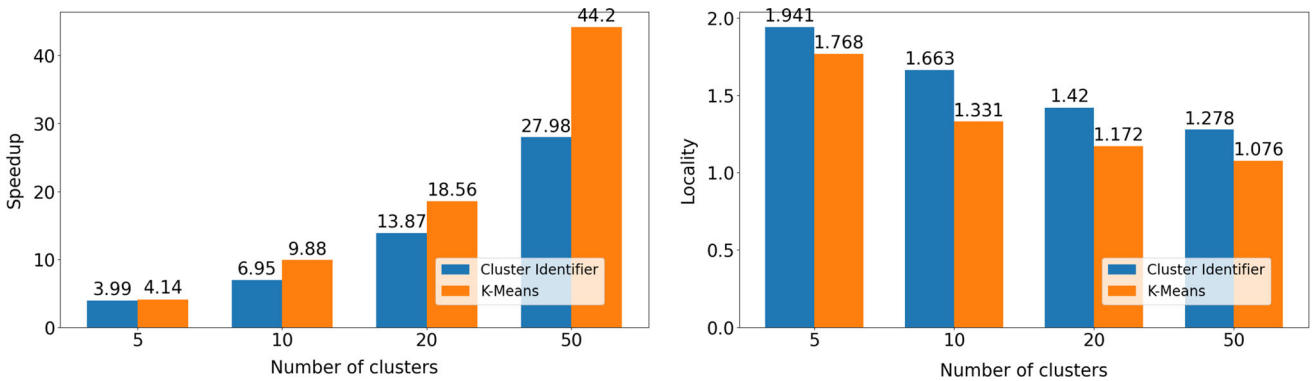

Fig. 4 Speedup and Locality of the two approaches. *Cluster Identifier* denotes our main approach, whereas *K-Means* denotes the alternative based on K-means

used metrics while possessing the two aforementioned properties. In particular, we found a significant speed-up that scales well with the number of clusters, as well as a within conversation locality well-above a traditional clustering setup. Moreover, we experimented with knowledge distillation and multitask learning, that can provide additional benefits to the final system.

Acknowledgements This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project #A18A2b0046).

References

1. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In Advances in neural information processing systems, pp 5998–6008
2. Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805
3. Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding by generative pre-training. https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/languageunderstandingpaper.pdf
4. Minaee S, Kalchbrenner N, Cambria E, Nikzad N, Chenaghlu M, Gao J (2021) Deep learning based text classification: a comprehensive review. ACM Comput Surv 54
5. Cambria E, Li Y, Xing F, Poria S, Kwok K (2020) SenticNet 6: ensemble application of symbolic and subsymbolic AI for sentiment analysis. In: CIKM, pp 105–114
6. Fadel A, Al-Ayyoub M, Cambria E (2020) JUSTers at SemEval-2020 Task 4: evaluating transformer models against commonsense validation and explanation. In: Proceedings of the fourteenth workshop on semantic evaluation, international committee for computational linguistics, Barcelona, pp 535–542
7. Mehta Y, Majumder N, Gelbukh A, Cambria E (2020) Recent trends in deep learning based personality detection. Artif Intell Rev 53:2313–2339
8. Ragusa E, Gianoglio C, Zunino R, Gastaldo P (2020) Image polarity detection on resource-constrained devices. IEEE Intell Syst 35(6):50–57
9. Han S, Mao H, Dally WJ (2015) Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv preprint arXiv:1510.00149
10. McDanel B, Teerapittayanon S, Kung HT (2017) Embedded binarized neural networks. arXiv preprint arXiv:1709.02260
11. Duong L, Hoang VCD, Pham TQ, Hong Y-H, Dovgalecs V, Bashkansky G, Black J, Bleeker A, Le Huitouze S, Johnson M (2019) An adaptable task-oriented dialog system for stand-alone embedded devices. In: Proceedings of the 57th annual meeting of the association for computational linguistics: system demonstrations, pp 49–57
12. Hesamifard E, Takabi H, Ghasemi M, Jones C (2017) Privacy-preserving machine learning in cloud. In: Proceedings of the 2017 on cloud computing security workshop, pp 39–43
13. Arora S, Batra K, Singh S (2013) Dialogue system: a brief review. arXiv preprint arXiv:1306.4134
14. Xu H, Peng H, Xie H, Cambria E, Zhou L, Zheng W (2020) End-to-end latent-variable task-oriented dialogue system with exact log-likelihood optimization. World Wide Web 23:1989–2002
15. Young T, Cambria E, Chaturvedi I, Zhou H, Biswas S, Huang M (2018) Augmenting end-to-end dialogue systems with commonsense knowledge. In AAAI, pp 4970–4977
16. Young T, Pandelea V, Poria S, Cambria E (2020) Dialogue systems with audio context. Neurocomputing 388:102–109
17. Ma Y, Nguyen KL, Xing F (2020) A survey on empathetic dialogue systems. Inf Fusion 64:50–70
18. Li W, Shao W, Ji S, Cambria E (2020) BiERU: bidirectional emotional recurrent unit for conversational sentiment analysis. arXiv preprint arXiv:2006.00492
19. Lowe R, Pow N, Serban I, Pineau J (2015) The ubuntu dialogue corpus: a large dataset for research in unstructured multi-turn dialogue systems. arXiv preprint arXiv:1506.08909
20. Zhou X, Dong D, Wu H, Zhao S, Yu D, Tian H, Liu X, Yan R (2016) Multi-view response selection for human–computer conversation. In: Proceedings of the 2016 conference on empirical methods in natural language processing, pp 372–381
21. Boussaha BEA, Hernandez N, Jacquin C, Morin E (2019) Deep retrieval-based dialogue systems: a short review. arXiv preprint arXiv:1907.12878
22. Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531
23. Humeau S, Shuster K, Lachaux M-A, Weston J (2019) Real-time inference in multi-sentence tasks with deep pretrained transformers. arXiv preprint arXiv:1905.01969
24. Ruder S (2017) An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098
25. Mazaré P-E, Humeau S, Raison M, Bordes A (2018) Training millions of personalized dialogue agents. arXiv preprint arXiv:1809.01984
26. Yang L, Qiu M, Qu C, Guo J, Zhang Y, Croft WB, Huang J, Chen H (2018) Response ranking with deep matching networks and external knowledge in information-seeking conversation systems. In: The 41st international ACM SIGIR conference on research & development in information retrieval, pp 245–254
27. Vig J, Ramea K (2019) Comparison of transfer-learning approaches for response selection in multi-turn conversations. In Workshop on DSTC7
28. Shrivastava A, Li P (2014) Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). In Advances in neural information processing systems, pp 2321–2329
29. Neyshabur B, Srebro N (2014) On symmetric and asymmetric lshs for inner product search. arXiv preprint arXiv:1410.5518
30. Bachrach Y, Finkelstein Y, Gilad-Bachrach R, Katzir L, Koenigstein N, Nice N, Paquet U (2014) Speeding up the xbox recommender system using a Euclidean transformation for inner-product spaces. In: Proceedings of the 8th ACM conference on recommender systems, pp 257–264
31. Malkov Y, Ponomarenko A, Logvinov A, Krylov V (2014) Approximate nearest neighbor algorithm based on navigable small world graphs. Inf Syst 45:61–68
32. Malkov YA, Yashunin DA (2018) Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans Pattern Anal Mach Intell 42:824–836
33. Zhang M, Wang W, Liu X, Gao J, He Y (2018) Navigating with graph representations for fast and scalable decoding of neural language models. In Advances in neural information processing systems, pp 6308–6319
34. Guo R, Kumar S, Choromanski K, Simcha D (2016) Quantization based fast inner product search. In Artificial intelligence and statistics, pp 482–490
35. Wu X, Guo R, Suresh AT, Kumar S, Holtmann-Rice DN, Simcha D, Yu F (2017) Multiscale quantization for fast similarity search. In Advances in neural information processing systems, pp 5745–5755

36. Yu H-F, Hsieh C-J, Lei Q, Dhillon IS (2017) A greedy approach for budgeted maximum inner product search. In Advances in neural information processing systems, pp 5453–5462
37. Chen PH, Si S, Kumar S, Li Y, Hsieh C-J (2018) Learning to screen for fast softmax inference on large vocabulary neural networks. arXiv preprint arXiv:1810.12406
38. Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078
39. Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In Empirical methods in natural language processing (EMNLP), pp 1532–1543
40. Shaham U, Stanton K, Li H, Nadler B, Basri R, Kluger Y (2018) Spectralnet: spectral clustering using deep neural networks. arXiv preprint arXiv:1801.01587
41. Tang R, Lu Y, Liu L, Mou L, Vechtomova O, Lin J (2019) Distilling task-specific knowledge from bert into simple neural networks. arXiv preprint arXiv:1903.12136
42. Li Y, Su H, Shen X, Li W, Cao Z, Niu S (2017) Dailydialog: a manually labelled multi-turn dialogue dataset. arXiv preprint arXiv:1710.03957
43. Ritter A, Cherry C, Dolan B (2010) Unsupervised modeling of twitter conversations. In: Human language technologies: the 2010 annual conference of the North American chapter of the association for computational linguistics, pp 172–180
44. Chen Q, Wang W (2019) Sequential attention-based network for noetic end-to-end response selection. arXiv preprint arXiv:1901.02609
45. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980