Retrieval-Augmented Transformer-XL for Close-Domain Dialog Generation

Giovanni Bonetta,[1,2] **Rossella Cancelliere,**[1] **Ding Liu,**[2] **Paul Vozila**[2]
[1]Department of Computer Science, University of Turin, Torino, Italy
[2]Nuance Communications Inc., Burlington, MA, USA
{giovanni.bonetta, rossella.cancelliere}@unito.it, {ding.liu, paul.vozila}@nuance.com

Abstract

Transformer-based models have demonstrated excellent capabilities of capturing patterns and structures in natural language generation and achieved state-of-the-art results in many tasks. In this paper we present a transformer-based model for multi-turn dialog response generation. Our solution is based on a hybrid approach which augments a transformer-based generative model with a novel retrieval mechanism, which leverages the memorized information in the training data via k-Nearest Neighbor search. Our system is evaluated on two datasets made by customer/assistant dialogs: the Taskmaster-1, released by Google and holding high quality, goal-oriented conversational data and a proprietary dataset collected from a real customer service call center. Both achieve better BLEU scores over strong baselines.

Introduction

Automatic dialog generation is become today a fundamental component for many real-world, challenging applications, such as virtual assistants, chatbots, etc., and is also a matter of great concern for companies and organizations relying on artificial intelligence solutions to enhance millions of daily interactions through their services.

Simple single-turn Seq2Seq architectures, initially proposed for this task, often fail to capture long-term temporal dependencies across dialog turns. (Sutskever, Vinyals, and Le 2014; Vinyals and Le 2015; Li et al. 2016). Multi-turn Seq2Seq models, such as the hierarchical recurrent encoder decoder (HRED) (Serban et al. 2016; Xing et al. 2018; Serban et al. 2017) have tried to alleviate these problems, yielding responses more coherent with the dialog contexts. Nonetheless, the generated texts tend to be either generic or too short, and not comparable with the human ones. Recently, pretrained transformer-based models such as BERT (Devlin et al. 2018), Transformer-XL (Dai et al. 2019), XL-Net (Yang et al. 2019) and ERNIE (Zhang et al. 2019) led to state-of-the-art performance on many natural language processing/understanding (NLP/NLU) tasks, including question answering, sentence classification, sentence similarity inference, and named entity recognition etc.

An interesting idea which further enhances the generative model performance is to condition the generation on samples retrieved from a task-related datastore. In (Guu et al. 2020; Lee, Chang, and Toutanova 2019) a generative model is augmented with a neural retriever trained to pick informative text paragraphs; (Khandelwal et al. 2020) propose to enhance a language model (LM) through a nearest neighbor search in suitable text collections. The model we present in this paper exploits a similar framework for dialog generation. Our first original contribution is showing how to generate dialog continuations using a LM augmented with a k-nearest neighbors (kNN) based retrieval mechanism. Furthermore, we exploit the typical dialog structure to enhance and speed the retrieval mechanism, improving the generation results. In section "Model Overview" we introduce our model and formally define our approach, also going into detail of the retrieval mechanism. The remaining sections are devoted to the dataset descriptions and results discussion.

Model Overview

We propose a method which improves dialog generation by exploiting memorized information from the training data, without further model training. At inference, turn generation is enhanced by interpolating the next word distribution based on the trained LM with the one based on a kNN search system. A single LM forward pass over the training data is preliminary conducted to compute context-target pairs and store them in a key-value pair *datastore*, which will be queried to perform the kNN search. The next sections describe this procedure and how a kNN distribution is computed and used to augment the LM.

Datastore Creation

The first step in order to create the datastore is the training of a LM, in our case a Transformer-XL (Dai et al. 2019), by minimizing the cross entropy of the training data. Overfitting is controlled through early stopping on validation data performance. Differently from (Dai et al. 2019) and (Khandelwal et al. 2020), which train a LM by concatenating all the examples, we train the model by resetting the Transformer-XL states at the beginning of each chat: this effectively prevents the model from conditioning on previous unrelated

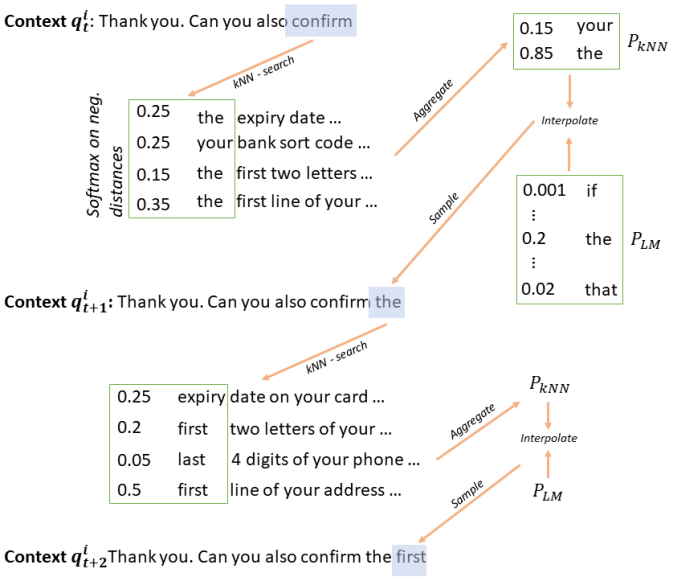

Figure 1: Illustration of the Generation Process

contexts.

Let $(c_t^i, w_t^i) \in D$ be the i^{th} example in training data D. The context c_t^i is a sequence of dialog turns of a dyadic chat occurring between an assistant and a user; c_t^i is represented as a sequence of tokens, i.e. $c_t^i = (w_1^i, w_2^i \dots w_{t-1}^i)$, and w_t^i is the target word.

Let $f(c_t^i)$ denote the context-encoder function, that maps the context c_t^i to its fixed-length vector embedding. We define $f(\cdot)$ as the input to the last feedforward layer in the final attention block of Transformer-XL, as in (Khandelwal et al. 2020). This achieves better performance than other options (e.g, the output of the last transformer layer). More specifically, $f(c_t^i)$ represents the embedding of token w_{t-1}^i after attending to all the previous tokens in the example.

Through one forward pass on the training data, the trained LM is used to build the datastore (K, W) containing the embeddings of all the tokens in the training data:

$$(K, W) := (k_t^i, w_t^i) = (f(c_t^i), w_t^i), \quad \forall (c_t^i, w_t^i) \in D$$

where $k_t^i = f(c_t^i)$ is the vector representation of the context, and w_t^i is the target word id (i.e. integer number).

Hybrid Probability Distribution

At inference, at every time step t, the trained LM receives a *query* (q_t), i.e. a chat truncated at the end of a user turn, and generates the next assistant turn token-by-token, according to the following steps, also illustrated in Fig. 1:

- Generate the context embedding $f(q_t)$ and the probability distribution $P_{LM}(v_t|q_t)$ over next words in the vocabulary
- Issue a kNN search with $f(q_t)$ as query, to get from the datastore its nearest neighbors N_t:

$$N_t = \{(k_1, w_1), (k_2, w_2) \dots (k_n, w_n) \dots\}$$

- Compute the score $S_{kNN}(w_n|q_t)$ of the token w_n over N_t, based on L^2 distance between k_n and $f(q_t)$:

$$S_{kNN}(w_n|q_t) = \frac{e^{-d(k_n, f(q_t))}}{\sum_{k_j \in N_t} e^{-d(k_j, f(q_t))}}$$

- Aggregate the scores of each vocabulary token w_n as the sum of all its occurrences within the retrieved neighbors:

$$S_{kNN}^{Aggr}(w_n|q_t) = \sum_{w_{n'} \in N_t w_{n'} = w_n} S_{kNN}(w_{n'}|q_t)$$

- Get the probability distribution P_{kNN} over next words in the vocabulary:

$$P_{kNN}(v_t|q_t) = \sum_{(k_n, w_n) \in N_t} \mathbf{1}_{v_t = w_n}(S_{kNN}^{Aggr}(w_n|q_t))$$

where $\mathbf{1}_{v_t=w_n}$ is a vector whose dimension is equal to the vocabulary size and whose elements are all zero except for the t-th one, equal to 1.

- Interpolate P_{kNN} with P_{LM} to get the final probability distribution P for next word v_t :

$$P(v_t|q_t) = \lambda P_{kNN}(v_t|q_t) + (1-\lambda) P_{LM}(v_t|q_t)$$

- Sample the next word $\hat{v}_t$ by greedily sampling from $P(v_t|q_t)$ and concatenate $\hat{v}_t$ to q_t to update the context: $q_{t+1} = q_t + \hat{v}_t$

If $\hat{v}_t$ is a terminal token the generation process stops; otherwise the entire procedure is repeated.

Retrieval Mechanism

To search the datastore, we use FAISS (Johnson, Douze, and Jégou 2017), an open source library for fast nearest neighbor retrieval in high dimensional space. FAISS's central building block is the *index*, a structure which stores millions of key-value pairs for efficient search. An issue with the index is that the number of elements could easily grow to hundreds of millions, leading to memory issues and hindering the search performance. However in practice, we only need to store token embeddings for assistant turns, since we are only interested in generating assistant responses. So we propose the simple but effective idea of filtering out from the datastore every token coming from a user turn, so almost halving its size, and allows the generation of consistent utterances, resembling assitant specific style.

Dataset Description

Two different datasets are used as benchmarks for our method: a public dataset, the Taskmaster-1, released by Google in 2019 and a real, company collected, call center customer service dataset.

Taskmaster-1 dataset. Taskmaster-1 (Byrne et al. 2019) is a crowsurced dataset, where Amazon turkers were asked to write dyadic dialogs following some given set of instructions describing six tasks: ordering pizza, creating auto repair appointments, setting up rides for hire, ordering movie tickets, ordering coffee drinks and making restaurant reservations.

Table 1: Dataset specifications.

	Taskmaster-1	**Prop. dataset**
# dialogs	7,708	1,328,301
# turns	169,467	21,953,321
# unique tokens	29,626	1,601,647
avg. turn per chat	21.99	16.53
avg. tokens per turn	7.83	18.00

Workers were asked to play the role of both assistant and user. Specifically, they were told to write a scenario in which they are speaking to their assistant on the phone while the assistant accesses the services for one of the given tasks. The resulting dataset contains 7,708 conversations. More info about the dataset are in table 1.

Proprietary (Prop.) dataset.[1] This dataset contains dyadic agent-user chats collected from a financial service call center over a one year time period, giving us the opportunity to test our approach in a real company scenario. It contains 172 times the dialogs number of the Taskmaster-1, as shown in table 1, and comes with two meta-information, the turn numbers and the agent-ids. The turn number is just the position of the specific turn within the chat, while the agent-id is a unique identifier for the agent speaking. We concatenate these information to the chat's text, following the approach used in (Wolf et al. 2019). An example is given in figure 3.

Implementation Details and Results

In this section we present the model implementation details and discuss the results obtained for both datasets.

Taskmaster-1 dataset

For the Taskmaster-1 we used a Transformer-XL model with 12 layers, 8 heads, 512-dimensional hidden states and 2048 as inner attention dimension, resulting in 49M weights and trained for a maximum of 10k steps optimizing with Adam. The training stopping criterion is based on perplexity on the development set. Hyperparameter tuning, including optimal λ determination, is done through performance evaluation over the development set. We adopted a BPE vocabulary (Sennrich, Haddow, and Birch2015) consisting of 16K tokens and generated using the Sentencepiece library (Kudo and Richardson 2018). All the training set is used to build the datastore.

Our model Transformer-XL + kNN is compared with two baselines: -Transformer, the best performing model by (Byrne et al. 2019) and - Transformer-XL, i.e. the LM used without the retrieval mechanism. The first column of table 2 shows the corresponding BLEU scores[2], obtained as mean values of 10 different runs, and standard deviations. We can see that our method gets more than two BLEU points over the Transformer baseline, and more than one point over the Transformer-XL baseline.

[1]The dataset can not be made public due to privacy constraints

[2]BLEU script at: https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/bin/t2t-bleu

Table 2: Average BLEU and standard deviations on test set. The statistical significance is validated via Student's t-test with significance level of 99.8%.

	Taskmaster-1		**Prop. dataset**	
Models:	Avg	Std	Avg	Std
Transf. (Byrne et al. 2019)	6.11[3]	-	-	-
Transf.-XL	7.09	0.14	39.96	0.36
Transf.-XL + kNN	**8.30**	0.05	**41.72**	0.20

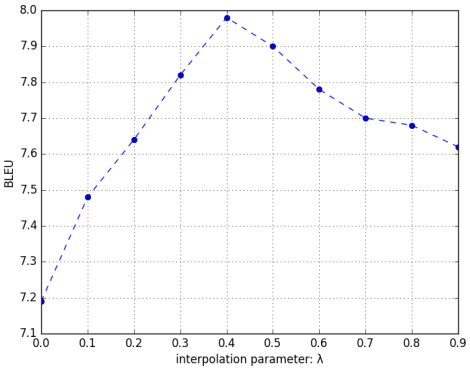

Figure 2: Taskmaster-1 BLEU trend (development set).

Figure 2 depicts the BLEU trend curve when the interpolation parameter λ varies through the selected range. We can see that kNN interpolation improves the BLEU scores over the Transformer-XL baseline for every value of λ in the selected range. The best result is with $\lambda = 0.4$, indicating LM and context retrieval are almost equally contributing.

Proprietary dataset

For the proprietary dataset we used the same model hyperparameters as for the Taskmaster-1 but augmented the hidden states dimension to 768 and the inner attention dimension to 3072, resulting in ˜116M weights. We trained for a maximum of 400k steps.

Since using all the training set for the datastore would result in a prohibitively large disk space usage we decided to build it using just the last 3 months of the training set ($1/4$ of the entire data). This resulted in ˜176M embeddings which occupy ˜500GB of disc memory. Also in this case the Transformer-XL + kNN improves over the LM model for about 1.8 BLEU points, even with a datastore smaller then the entire training set. These results are obtained interpolating with $\lambda = 0.5$ (best on dev. set).

Figure 3 shows a sample from the test data along with the expected target, the turn generated by the Transformer-XL, and the turn generated by our Transformer + kNN. In this dialog a user wants some help for a credit card application. Our proposed model generates a sensible and relevant

[3]Results from original paper

QUERY:

...
<AGENT> 3 agent@company.com | I am thrilled to hear that you would like to apply for a new card today. </AGENT>
<AGENT> 3 agent@company.com | Have you decided which card you would like to apply for? </AGENT>
<USER> 3 | yes </USER>
<AGENT> 4 agent@company.com | Excellent! </AGENT>
<USER> 4 | basic credit card </USER>
<AGENT> 5 agent@company.com |

TARGET:
Our ... is a great card for ... and balance transfers. You can use the link below to begin the application today. </AGENT>

Transformer-XL:
That is very good to know. </AGENT>

Transformer-XL + kNN:
That is a great card. I would be happy to stand by while you apply should you have additional questions. </AGENT>

Figure 3: Example of inference query, along with results from baseline and our best model. *agent@company.com* is the agent-id, which is preceded by the turn number. Tokens between angular parenthesis indicate the beginning and end of turns.

continuation: the agent conveys the intent to help the user apply for the credit card, as in the target. On the other hand the baseline Transformer-XL model generates a generic response which is not useful in advancing the dialog.

Conclusions

In this work we shown how to enhance a generative model for dialog completion by pairing it with an effective retrieval system. Our approach achieves higher BLEU scores than strong generative models when tested on two challenging datasets. Moreover, our solution often outputs more sensible/informative dialog turns. In the future we plan to extend this preliminary work analysing more models on different datasets, and further investigating results and generated examples.

References

Byrne, B.; Krishnamoorthi, K.; Sankar, C.; Neelakantan, A.; Goodrich, B.; Duckworth, D.; Yavuz, S.; Dubey, A.; Kim, K.; and Cedilnik, A. 2019. Taskmaster-1: Toward a realistic and diverse dialog dataset. In Inui, K.; Jiang, J.; Ng, V.; and Wan, X., eds., *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, 4515–4524. Association for Computational Linguistics.

Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J. G.; Le, Q. V.; and Salakhutdinov, R. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR* abs/1901.02860.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805.

Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; and Chang, M. 2020. REALM: retrieval-augmented language model pre-training. *CoRR* abs/2002.08909.

Johnson, J.; Douze, M.; and Jégou, H. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.

Khandelwal, U.; Levy, O.; Jurafsky, D.; Zettlemoyer, L.; and Lewis, M. 2020. Generalization through memorization: Nearest neighbor language models. In *Proceedings of the 2020 International Conference on Learning Representations*.

Lee, K.; Chang, M.; and Toutanova, K. 2019. Latent retrieval for weakly supervised open domain question answering. *CoRR* abs/1906.00300.

Li, J.; Galley, M.; Brockett, C.; Gao, J.; and Dolan, B. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 110–119. San Diego, California: Association for Computational Linguistics.

Serban, I.; Sordoni, A.; Bengio, Y.; Courville, A.; and Pineau, J. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI 2016)*, 3776—-3784.

Serban, I. V.; Klinger, T.; Tesauro, G.; Talamadupula, K.; Zhou, B.; Bengio, Y.; and Courville, A. 2017. Multiresolution recurrent neural networks: An application to dialogue response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*.

Sutskever, I.; Vinyals, O.; and Le, Q. 2014. Sequence to sequence learning with neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 3104—3112.

Vinyals, O., and Le, Q. 2015. A neural conversational model. In *Proceedings of ICML Deep Learning Workshop*.

Wolf, T.; Sanh, V.; Chaumond, J.; and Delangue, C. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *CoRR* abs/1901.08149.

Xing, C.; Wu, Y.; Zhou, M.; Huang, Y.; and Ma, W.-Y. 2018. Hierarchical recurrent attention network for response generation. In *Proceedings of the The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, 5610—-5617.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J. G.; Salakhutdinov, R.; and Le, Q. V. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR* abs/1906.08237.

Zhang, Z.; Han, X.; Liu, Z.; Jiang, X.; Sun, M.; and Liu, Q. 2019. ERNIE: enhanced language representation with informative entities. *CoRR* abs/1905.07129.