Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 735–742
Marseille, 11–16 May 2020

Which Model Should We Use for a Real-World Conversational Dialogue System? a Cross-Language Relevance Model or a Deep Neural Net?

Seyed Hossein Alavi, Anton Leuski, David Traum
USC Institute for Creative Technologies
12015 E Waterfront Dr, Los Angeles, CA 90094
{seyedhoa, leuski, traum}@ict.usc.edu

Abstract

We compare two models for corpus-based selection of dialogue responses: one based on cross-language relevance with a cross-language LSTM model. Each model is tested on multiple corpora, collected from two different types of dialogue source material. Results show that while the LSTM model performs adequately on a very large corpus (millions of utterances), its performance is dominated by the cross-language relevance model for a more moderate-sized corpus (ten thousands of utterances).

Keywords: Deep Neural Network, Dual-Encoder model, Cross-Language Relevance model, Ubuntu dialogue corpus, end-to-end conversational dialogue

1. Introduction

End-to-end neural network models of conversational dialogue have become increasingly popular for conversational tasks (Ritter et al., 2011; Serban et al., 2015; Zhao et al., 2017). These models eschew traditional dialogue modeling approaches that include internal hand-crafted domain models and representations of dialogue context and separate components for understanding natural language (converting to the internal representation language), updating dialogue state, state-based response generation, and natural language generation (Traum and Larsson, 2003; Raux et al., 2005). Instead, these models learn to respond directly from a corpus, either by generating new responses or selecting a response from the corpus training data, using dual encoding and hidden layers to learn appropriate dialogue continuations. However, there are still a number of questions remaining about how well such models really work for real applications, and how much data is needed to achieve acceptable performance. Other machine learning approaches have been shown to be useful, with much smaller data sets.

In this paper, we compare two different kinds of end-to-end system, a neural network model based on (Lowe et al., 2015) and an older kind of end-to-end dialogue model, based on cross-language retrieval (Leuski et al., 2006), implemented in the publicly available NPCEditor (Leuski and Traum, 2011), and previously used for systems that have been displayed in museums (Traum et al., 2012; Traum et al., 2015). We compare these models on two different datasets: the Ubuntu Corpus (Lowe et al., 2015), and one derived from one of the museum system datasets (Traum et al., 2015). In the next section, we describe the data sets we use for the experiments. In section 3 we describe the NPCEditor and its selection model. We then review some prior neural network approaches to dialogue in section 4, followed by details of the model we tested in section 5. In section 6, we describe the experiments and results. Finally, we conclude in section 7 with a discussion of the results and their implications.

2. Datasets

We utilized four datasets in our experiments to compare NPCEditor with a deep neural network model. The *Ubuntu Dialogue corpus* (Lowe et al., 2015) was constructed from Linux support message boards, where people posted problems and solutions. We constructed three other datasets out of the data made available from the system described in (Traum et al., 2015), in which people ask questions of a Holocaust survivor, and receive recorded responses.

2.1. Ubuntu Dialogue Corpus

As a first baseline, we use the *Ubuntu Dialogue corpus* which contains 1 million multi-turn dialogues, with a total of over 7 million utterances and 100 million words. The training set has 50% positive and 50% negative pairs of <context, response>. In the development, for a given context it has 1 relevant response and 9 distractors. This corpus has previously been used for testing neural network end-to-end dialogue systems.

2.2. Pinchas data

The initial data that was used for training the system in (Traum et al., 2015), consists of 23830 pairs of questions and responses from the interviews. We divided this set into three subsets: 70% for the training set (16675 samples), 20% for development set (4764 samples), and 10% percent for the test set (2383 samples). Please notice that the mentioned subsets only contained positive samples. We constructed a set of negative samples by removing positive samples from all possible < question, response > pairs.

2.2.1. Pinchas_10

We doubled the amount of training set by adding randomly selected negative samples to it. In the end, we came up with 33350 samples for the training set, 50% of which were negative samples and the rest were positive ones. For each positive sample in the development and test sets, we added nine distractors. Distractors were randomly selected from the pool of invalid responses for the given context. So each entry of the pinchas_10 development and test sets would be of the form:

Context	Response	Label
where do you live now	i live in toronto canada	1
how do you define love	when i arrived in england first i was in windermere in a little place cal	0
did you play any sports	um i did not play sports as a child i didn't get any opportunity	1
can you tell anything about your twin sister	what i remember about my sister are actually three things but i can't	1
can you share with me any music remember you	my best memory from before the war was um i spent i was ill at one s	0

Figure 1: A subset of Pinchas training set that we constructed similar to the structure of Ubuntu Dialogue corpus training set. Label "0" means context and response are not relevant and Label "1" means they are relevant.

$$< question, response, d_1, d_2, \ldots, d_9 >$$

2.2.2. Pinchas_1444_v1

We constructed Pinchas_1444 to investigate how the models would perform on a task inspired by a real problem. Our goal was to create a system that could return an appropriate response from a set of all 1444 available responses.
For the training set, we followed the same procedure that we did in constructing Pinchas_10. Nonetheless, for the development and test sets, instead of 10 distractors, we used the whole set of possible responses. Another important difference between Pinchas_1444 and Pinchas_10 is that in this new set there might be more than one relevant response for a given question.

2.2.3. Pinchas_1444_v2

After constructing Pinchas_1444_v1, we decided to construct another dataset in order to test whether increasing negative samples in the training set would influence the results. Given that very few of the 1444 responses are appropriate for any given question, showing an even number of positive and negative examples might inappropriately prefer recall over precision. We increased the proportion of negative samples from 50% to 90% in the training set. We accomplished this by adding more negative samples to the Pinchas_1444_v1.
Each entry of the Pinchas_1444_v1 and Pinchas_1444_v2 development and test sets is of the form:

$$< question, r_1, r_2, ..., r_{1444} >$$

3. NPCEditor

The first model we are testing is NPCEditor (Leuski and Traum, 2011), which was used for the system in (Traum et al., 2015). At the core of NPCEditor is a statistical regression approach based on cross-lingual language model proposed by Lavrenko for cross-lingual information retrieval (Lavrenko, 2004). Leuski and Traum successfully adopted his approach to question answering and applied it in many different applications (Leuski and Traum, 2008; Leuski and Traum, 2011). We were interested in this approach as a baseline for our experiments because it seems to be robust and has a small number parameters to tune. The NPCEditor is publicly availible as part of the Virtual Human Toolkit (Hartholt et al., 2013).
Each text utterance in the collection, – including both questions and answers, – is represented by a language model or a probability distribution $P(w)$ over words in the vocabulary. It is assumed that vocabularies of questions and answers are different, so it is not possible to compare a language model $P_Q(w)$ of a question Q directly to a language model $P_A(w)$ of an answer A. Instead, a set of training question-answer pairs is used to estimate a cross-language relevance model $P(w_A|Q)$ – a conditional probability of observing a word w_A in an answer in response to a question Q. This estimate is

$$P(w_A|Q) = \frac{\sum_{(q,a)} p_a(w_A) \prod_{v \in Q} p_q(v)}{\sum_{(q,a)} \prod_{v \in Q} p_q(v)} \quad (1)$$

Here (q, a) indicates a training question-answer pair, $p_a(w_A)$ is the probability estimation of observing w_A in answer a, $p_q(v)$ is the probability estimation of observing a question word v from Q in the training question q. Maximum Likelihood Estimation (MLE) is used for both $p_a(w_A)$ and $p_q(v)$:

$$p_S(w) = \lambda \frac{\#(w, S)}{|S|} + (1 - \lambda) \frac{\sum_{\forall s} \#(w, s)}{\sum_{\forall s} |s|} \quad (2)$$

Here $\#(w, S)$ is the number of times a word w appears in utterance S, $|S|$ is the total number of words in the utterance, and λ is a tunable parameter that can be determined from training. There are two λ parameters to tune, one for questions and one for answers. To compare language models Kullback-Leibler divergence is used:

$$D(Q||A) = \sum_a P(a|Q) \log \frac{P(a|Q)}{p_A(a)} \quad (3)$$

Here the difference between a question Q and an answer A is defined as a sum over all words a in the answer vocabulary. Given a new question Q, NPCEditor computes this value for each known A and returns the one with the lowest value $D(Q||A)$.

4. Previous Neural Net Models

Previously, most dialogue research had historically focused on structured slot-filling tasks. Few approaches were proposed, yet leveraging more recent developments in neural learning architecture, exploiting deep learning for end to end dialogue systems has become interestingly more popular.
(Ritter et al., 2011) presented a response generation model for Twitter data based on phrased-based Statistical Machine Translation. They viewed the response generation problem as a machine translation problem. This is shown to give superior performance to previous information retrieval (e.g. nearest neighbor) approaches.

Context	can you sing something for us
Response	um yeah i always have a song it's not actually a song well it is a song
D1	happy story of my childhood when i was a little boy uh my father used to get
D2	i was born in lodz in poland
D3	the conditions in the ghetto were you know unbelievable they they they were
D4	i think the biggest challenge was to learn the language because without the
D5	many people have that experience but i don't have it
D6	uh when the war started um we were all very afraid of what's going to happe
D7	my my my parents helped my mother's whole family who came running awa
D8	i started off by living in czechoslovakia for several months i was then in engl
D9	uh yes he's still alive

Figure 2: Sample of Pinchas_10 test set that Dual-Encoder model could recognize the relevant response.

(Serban et al., 2015) investigated the task of building open domain conversational dialogue systems based on large dialogue corpora using hierarchical recurrent encoder-decoder neural network. They showed how the model's performance can be improved by bootstrapping the learning from a larger question-answer pair corpus and from pre-trained word embeddings.

(Lowe et al., 2015) released the Ubuntu Dialogue corpus, A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. They also introduced a deep learning model named Dual-Encoder model, suitable for analyzing this dataset on the task of selecting the best next response. There has been efforts in using Ubuntu Dialogue corpus (Mehri and Carenini, 2017) and some of them even introduced new models (e.g. (Dong and Huang, 2018)) for the task of selecting the best next response.

BERT (Devlin et al., 2018) is another recent developed tool for getting sentence embeddings. It is the latest refinement of a series of neural models that make heavy use of pre-training and with the aid of transformers (Vaswani et al., 2017) has led to impressive gains in many natural language processing tasks.

(Zhang et al., 2018) suggested a retrieval-based response matching model for multi-turn conversation. They tried to consider interactions among previous utterances in their context modeling by introducing a deep aggregation model to form a fine-grained context representation. (Zhou et al., 2018) is one of the latest models proposed for the task of next best response selection. They examined matching a response with its multi-turn context using dependency information based entirely on attention.

(Shao et al., 2017) adopted a variation of sequence to sequence model to generate high quality responses on the single turn setting. (Olabiyi et al., 2018) proposed an adversarial learning approach to the generation of multi-turn dialogue responses. Their method, which is based on conditional generative adversarial networks, generalizes better than previous works using only log-likelihood criterion and generates longer and more informative responses.

Considering all the recent efforts for generating or selecting responses, for our task in this study which is selecting the best next response, Dual-Encoder model, as one of the core deep neural net models, would be a good candidate to be representative of the cluster of neural net models introduced for this task.

5. Deep Neural Net Model

The NPCEditor model is a retrieval based model and it has shown to be functional on practical problems. In order to compare its performance with a deep neural network model, we trained a recurrent neural network with LSTM (Hochreiter and Schmidhuber, 1997) hidden units so for a given conversation and message, it predicts a matching score.

5.1. Dual-Encoder

First introduced by (Lowe et al., 2015), given context $C = (c_1, c_2, ..., c_i, ..., c_m)$ with length m, and response $R = (r_1, r_2, ..., r_j, ..., r_n)$ with length n where c_i and r_j are the ith and jth words in context and response, respectively, the model calculates a matching score $0 \leq s \leq 1$. $s = 1$ shows a perfect match and $s = 0$ means the response is irrelevant to the given context.

The Dual-Encoder model uses two recurrent neural networks with tied weights to calculate a score representing how much response R is a good match for context C. Figure 3 shows the architecture of the explained model. We used LSTM units as the RNN units for the hidden layer.

LSTM units can keep long-term dependencies due to their inner structure. The key to this LSTM capability is the internal state C_t. The LSTM does have the ability to remove or add information to the internal state through its so-called gates. C_t is a function of previous inner state C_{t-1}, previous hidden output h_{t-1}, and the given input x_t.

Given context C' and response R', first we compute their word embeddings using pre-trained vectors from GloVe (Pennington et al., 2014) and get C and R respectively. Then we feed the first encoder with C and the second encoder with R. Final hidden states in encoders, c and r, represent the summary of context C and response R. Using c and r we calculate a probability (score) indicating how much response R matches to context C:

$$s = p(flag = 1|c, r, M) = \sigma(c^T Mr + b)$$

Where bias b and matrix M are parameters learned during

the training set. Model is trained by minimizing the cross entropy of all labeled pairs:

$$L = -\sum_n logp(flag_n|c_n, r_n, M)$$

5.2. Implementation details

We train the model with the same parameters that (Lowe et al., 2015) did in their paper. We use a constant learning rate of 0.001. Batch size in the training and test is 512, however, for tuning on the dev set it is 256. Embedding dimension is 100 and hidden dimension is 300. For training on both versions of pinchas_1444 data, the best model is chosen based on the Rank whereas during training on pinchas_10 data the best model is chosen based on R@1.

6. Experiments and Evaluation

In this section, we introduce a series of experiments to compare the NPCEditor and the Dual-Encoder model described in the previous section. Following, (Lowe et al., 2015), we use R@k as the evaluation metric, which is the percentage of times that the expected response is retrieved in the top-k responses. R@1 is equivalent to accuracy. We first compare the Dual-Encoder model on both the Ubuntu corpus, to compare with the model in (Lowe et al., 2015), as a sanity check on the model, and on the Pinchas_10 dataset, which has a test-set parallel in structure to Ubuntu. Next we compare both the NPCEditor and the Dual-Encoder model on the Pinchas_10 dataset. Then we study the performance of Dual-Encoder model on Pinchas_1444_v1 and Pinchas_1444_v2. Finally, we compare the NPCEditor and the Dual-Encoder model (trained on two different data sets) on the more realistic Pinchas_1444 dataset.

6.1. Dual-Encoder on Pinchas_10 vs Dual-Encoder on Ubuntu

In the first experiment, we first examine the performance of Dual-Encoder model on the Ubuntu and Pinchas_10 datasets. Our goal is to investigate whether the proposed deep learning model returns similar or better results on a question-answering task (Pinchas data) compared to the Linux discussions in the Ubuntu corpus. From the results shown in Figure 4, we observe that the Dual-Encoder model works even better on the Pinchas_10 data than it does on the Ubuntu data. This is true for each of the window sizes for k in R@k that we tried. Results on the Ubuntu data set are, as expected, similar to those reported by (Lowe et al., 2015).

6.2. NPCEditor on Pinchas_10 vs Dual-Encoder on Pinchas_10

In the second experiment, our goal is to compare the performance of NPCEditor and Dual-Encoder model on the same task. The results from our experiment are available in Figure 5. Interestingly, on Pinchas_10 data, NPCEditor's performance is significantly better than Dual-Encoder model for R@1 which is returning only one response. INPCEditor and Dual-Encoder model get similar results for R@2 (0.84 and 0.83 respectively), however, Dual-Encoder model performs slightly better than NPC editor for R@5. Please notice that although NPCEditors performance for R@2 and R@5 is not better than Dual-Encoder model, at the end for completing the task of returning one response out of a set of possible responses, R@1 is what we are looking for and in this case, NPCEdiotr performs much better than Dual-Encoder model.

6.3. Increasing negative samples

In our third experiment, we compare Pinchas_1444_v1 and Pinchas_1444_v2 datasets, to examine the impact of increasing negative examples on Dual-Encoder's performance. Figure 6 shows that Dual-Encoder model performs better on the Pinchas_1444_v2 than Pinchas_1444_v1. This is true for each of the window sizes for k in R@k that we tried. The results are justifiable by considering the fact that although 90% of the training set in Pinchas_1444_v2 are negative samples, since there are more samples in the training set in comparison to Pinchas_1444_v1, the model learns more and achieves better results.

6.4. NPCEditor on Pinchas_1444 vs Dual-Encoder on Pinchas_1444

In our last experiment, we are interested to see how NPCEditor and the Dual-Encoder model perform on a real-world problem instead of artificially created tasks such as Ubuntu data and Pinchas_10. As described in section 2., Pinchas_1444 datasets are created to solve a real-world question answering problem. In the development and test sets of the mentioned datasets, for each context, we have a collection of 1444 possible responses to select from.
Furthermore, in some samples of Pinchas_1444, we may have more than one possible relevant response which is another difference between Pinchas_1444 and previous datasets. In such cases, it only matters for us that one or more of the relevant responses appear in the top-k responses to call the retrieval successful.
Since NPCEditor uses only positive examples in its training procedure and ignores the negative ones, it would not matter which version of Pinchas_1444 we use for its training. Thus we chose Pinchas_1444_v2 for training NPCEditor. The results of the experiment are shown in Figure 7. Interestingly results show that NPCEditor gets the accuracy (or R@1) of 0.76 on the Pinchas_1444_v2 data which is much better than Dual-Encoder model's performance on Pinchas_1444_v1 and Pinchas_1444_v2 (0.0625 and 0.1238 respectively). NPCEditor dominates Dual-Encoder for all of the values of k in this experiment.

7. Discussion and Future Directions

In order to complete the comparison, we wanted to also test NPCEditor on the Ubuntu corpus data. However the size and structure of the dataset made this challenging, so we defer this for future work. Experiment 1 showed that the Pinchas data appears easier than the Ubuntu data - with a much smaller training set size, the Dual-Encoder model was able to improve on R@k in the Pinchas_10 dataset compared to the Ubuntu dataset. Given the amount of available training data (10s of thousands of examples), the NPCEditor significantly out-performs the Dual-Encoder model in R@1 on this data set. The results are even more striking for a more real-world example, taken from (Traum et al., 2015), where

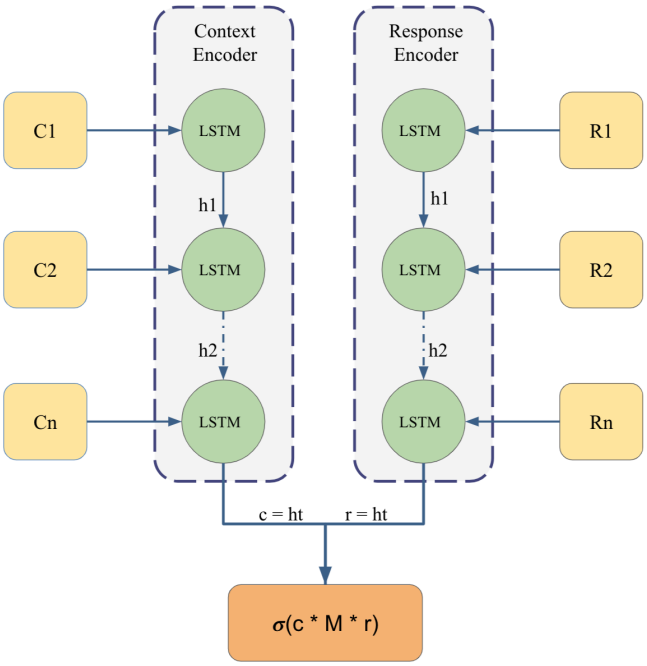

Figure 3: Dual-Encoder model. We considered contexts up to a maximum of t=160.

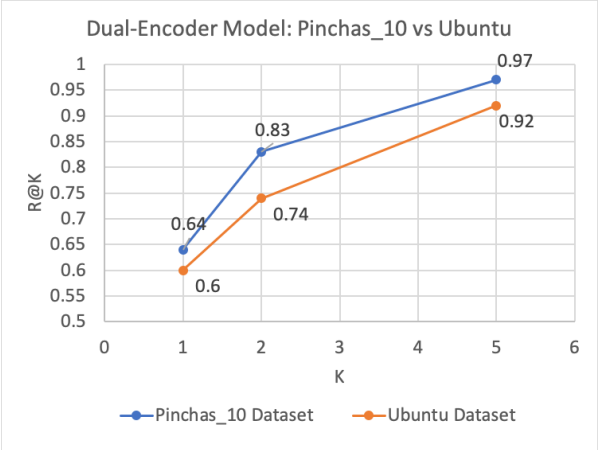

Figure 4: Results from experiment 1 using various R@k measures, which illustrates Dual-Encoder model works better on the Pinchas_10 data than Ubuntu data.

the system's task is to pick the best response out of a set of over 1000 available responses. Here, the Dual-Encoder model does not perform well enough to engage in a meaningful dialogue, while the NPCEditor performs similarly to results reported in (Traum et al., 2015), which led to much-reported user engagement. The improved performance of the Pinchas_1444_v2 training set, with a much higher proportion of negative examples, does perhaps point to a direction for improvement. Future work should perhaps look at the even higher distribution of negative to positive examples.

These results do show that despite the recent popularity of deep learning models, there is still a place for more traditional machine learning algorithms, that can operate well on more moderate-sized data sets for problems of interest. It may also be the case that different types of dialogue have different optimal models. For example, (Gandhe and Traum, 2010) show very different upper bounds for retrieval approaches to dialogue.

8. Acknowledgments

This work was supported by the U.S. Army. Any opinion, content or information presented does not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred. Parts of this work was appeared as an abstract paper in (Alavi et al., 2019).

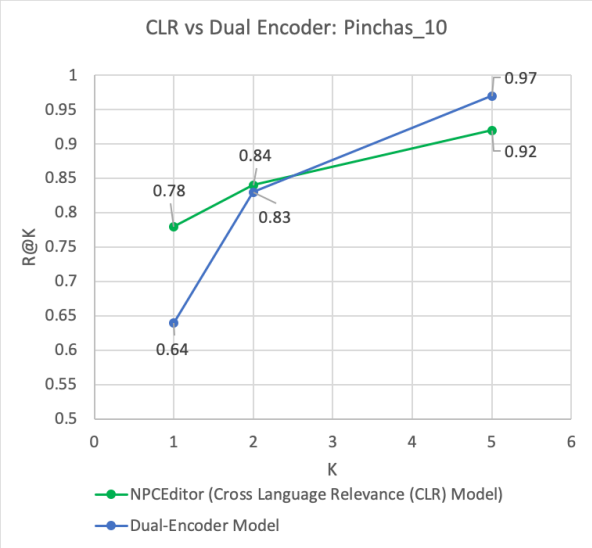

Figure 5: Results from experiment 2 using various R@k measures, which illustrates NPCEditor works better than the Dual-Encoder model for R@1 on the Pinchas_10 dataset, however, it has similar or slightly weaker results for R@2 and R@5.

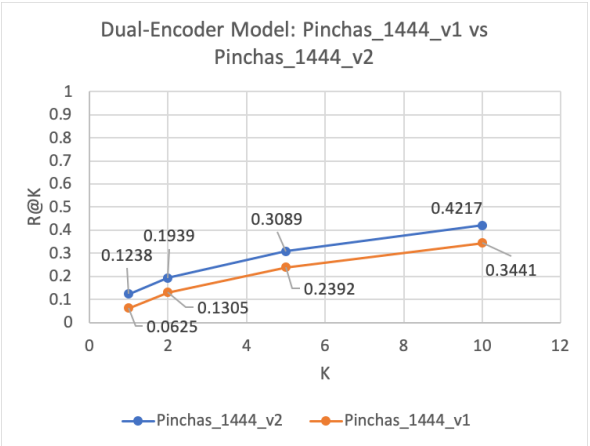

Figure 6: Results from experiment 3, which shows the Dual-Encoder model performs better on Pinchas_1444_v2 data.

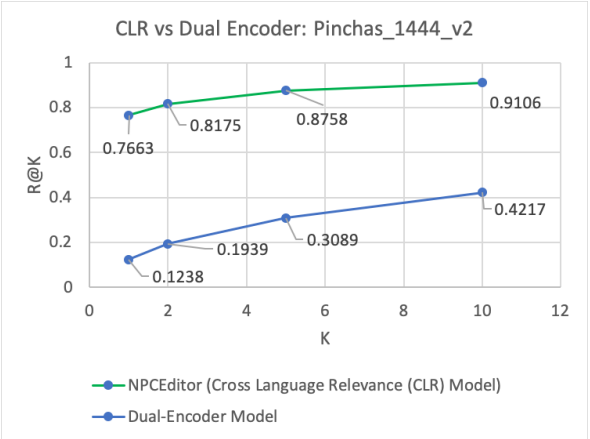

Figure 7: Results from experiment 4, which shows that Dual-Encoder model is dominated by NPCEditor on Pinchas_1444

Context	what's your favorite food
R1	no we never got any food any water except if we were taken into a bar
R2	el salvador
R3	um i had breakfast i had porridge for breakfast oatmeal i haven't had lu
R4	i had coke
R5	so i don't think you could because when you got a piece of bread and y

Figure 8: Top 5 responses returned by Dual-Encoder model for a sample of Pinchas_1444. For the given context, Dual-Encoder model failed to return a relevant response among top 10 returned responses.

Context	What's your favorite food
R1	my favorite food is gefilte fish that i make according to the way my mother used to make it
R2	um i don't remember having any favorite food i was a very healthy i had a very healthy appe
R3	my favorite festival was simchat torah is a rejoicing of the law the reason why is because i h

Figure 9: Top responses returned by NPCEditor model for a sample of Pinchas_1444. For the given context, NPCEditor model was successful to return relevant responses among top 3 returned responses.

9. References

Alavi, S. H., Leuski, A., and Traum, D. (2019). Comparing cross language relevance vs deep neural network approaches to corpus-based end-to-end dialogue systems. In *Proceedings of the 23rd Workshop on the Semantics and Pragmatics of Dialogue - Poster Abstracts*, London, United Kingdom, September. SEMDIAL.

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

Dong, J. and Huang, J. (2018). Enhance word representation for out-of-vocabulary on ubuntu dialogue corpus. *CoRR*, abs/1802.02614.

Gandhe, S. and Traum, D. (2010). I've said it before, and i'll say it again: an empirical investigation of the upper bound of the selection approach to dialogue. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 245–248. Association for Computational Linguistics.

Hartholt, A., Traum, D., Marsella, S. C., Shapiro, A., Stratou, G., Leuski, A., Morency, L.-P., and Gratch, J. (2013). All together now: Introducing the Virtual Human toolkit. In *International Conference on Intelligent Virtual Humans*, Edinburgh, UK, August.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Lavrenko, V. (2004). *A Generative Theory of Relevance*. Ph.D. thesis, University of Massachusetts at Amherst.

Leuski, A. and Traum, D. (2008). A statistical approach for text processing in virtual humans. In *Proceedings of the 26th Army Science Conference*, Orlando, Florida, USA, December.

Leuski, A. and Traum, D. (2011). NPCEditor: Creating virtual human dialogue using information retrieval techniques. *AI Magazine*, 32(2):42–56.

Leuski, A., Patel, R., Traum, D., and Kennedy, B. (2006). Building effective question answering characters. In *Proceedings of the 7th SIGdial Workshop on Discourse and Dialogue*, pages 18–27.

Lowe, R., Pow, N., Serban, I., and Pineau, J. (2015). The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *CoRR*, abs/1506.08909.

Mehri, S. and Carenini, G. (2017). Chat disentanglement: Identifying semantic reply relationships with random forests and recurrent neural networks. In *IJCNLP*.

Olabiyi, O., Salimov, A., Khazane, A., and Mueller, E. T. (2018). Multi-turn dialogue response generation in an adversarial learning framework. *CoRR*, abs/1805.11752.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Raux, A., Langner, B., Bohus, D., Black, A. W., and Eskenazi, M. (2005). Let's go public! taking a spoken dialog system to the real world. *Proceeding of the International Speech Communication Association*.

Ritter, A., Cherry, C., and Dolan, W. B. (2011). Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 583–593, Stroudsburg, PA, USA. Association for Computational Linguistics.

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2015). Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808.

Shao, L., Gouws, S., Britz, D., Goldie, A., Strope, B., and Kurzweil, R. (2017). Generating long and diverse responses with neural conversation models. *CoRR*, abs/1701.03185.

Traum, D. and Larsson, S. (2003). The information state

approach to dialogue management. In Jan van Kuppevelt et al., editors, *Current and New Directions in Discourse and Dialogue*, pages 325–353. Kluwer.

Traum, D., Aggarwal, P., Artstein, R., Foutz, S., Gerten, J., Katsamanis, A., Leuski, A., Noren, D., and Swartout, W. (2012). Ada and grace: Direct interaction with museum visitors. In *International conference on intelligent virtual agents*, pages 245–251. Springer.

Traum, D., Georgila, K., Artstein, R., and Leuski, A. (2015). Evaluating spoken dialogue processing for time-offset interaction. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 199–208, Prague, Czech Republic, September. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Zhang, Z., Li, J., Zhu, P., Zhao, H., and Liu, G. (2018). Modeling multi-turn conversation with deep utterance aggregation. *CoRR*, abs/1806.09102.

Zhao, T., Zhao, R., and Eskenazi, M. (2017). Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664, Vancouver, Canada, July. Association for Computational Linguistics.

Zhou, X., Li, L., Dong, D., Liu, Y., Chen, Y., Zhao, W. X., Yu, D., and Wu, H. (2018). Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1118–1127, Melbourne, Australia, July. Association for Computational Linguistics.