Self-Attentional Models Application in Task-Oriented Dialogue Generation Systems

Mansour Saffar Mehrjardi, Amine Trablesi, Osmar R. Zaïane
Department of Computing Science, University of Alberta
{saffarme,atrabels,zaiane}@ualberta.ca

Abstract

Self-attentional models are a new paradigm for sequence modelling tasks which differ from common sequence modelling methods, such as recurrence-based and convolution-based sequence learning, in the way that their architecture is only based on the attention mechanism. Self-attentional models have been used in the creation of the state-of-the-art models in many NLP tasks such as neural machine translation, but their usage has not been explored for the task of training end-to-end task-oriented dialogue generation systems yet. In this study, we apply these models on the three different datasets for training task-oriented chatbots. Our finding shows that self-attentional models can be exploited to create end-to-end task-oriented chatbots which not only achieve higher evaluation scores compared to recurrence-based models, but also do so more efficiently.

1 Introduction

Task-oriented chatbots are a type of dialogue generation system which tries to help the users accomplish specific tasks, such as booking a restaurant table or buying movie tickets, in a continuous and uninterrupted conversational interface and usually in as few steps as possible. The development of such systems falls into the Conversational AI domain which is the science of developing agents which are able to communicate with humans in a natural way (Ram et al., 2018). Digital assistants such as Apple's Siri, Google Assistant, Amazon Alexa, and Alibaba's AliMe are examples of successful chatbots developed by giant companies to engage with their customers.

There are mainly two different ways to create a task-oriented chatbot which are either using set of hand-crafted and carefully-designed rules or use corpus-based method in which the chatbot can be trained with a relatively large corpus of conversational data. Given the abundance of dialogue data, the latter method seems to be a better and a more general approach for developing task-oriented chatbots. The corpus-based method also falls into two main chatbot design architectures which are pipelined and end-to-end architectures (Chen et al., 2017). End-to-end chatbots are usually neural networks based (Shang et al., 2015; Dodge et al., 2015; Wen et al., 2016; Eric and Manning, 2017a) and thus can be adapted to new domains by training on relevant dialogue datasets for that specific domain. Furthermore, all sequence modelling methods can also be used in training end-to-end task-oriented chatbots. A sequence modelling method receives a sequence as input and predicts another sequence as output. For example in the case of machine translation the input could be a sequence of words in a given language and the output would be a sentence in a second language. In a dialogue system, an utterance is the input and the predicted sequence of words would be the corresponding response.

Self-attentional models are a new paradigm for sequence modelling tasks which differ from common sequence modelling methods, such as recurrence-based and convolution-based sequence learning, in the way that their architecture is only based on the attention mechanism. The Transformer (Vaswani et al., 2017) and Universal Transformer (Dehghani et al., 2018) models are the first models that entirely rely on the self-attention mechanism for both encoder and decoder, and that is why they are also referred to as a self-attentional models. The Transformer models has produced state-of-the-art results in the task neural machine translation (Vaswani et al., 2017) and this encouraged us to further investigate this model for the task of training task-oriented chatbots. While in the Transformer model there is no recurrence, it turns out that the recurrence used in RNN models is essential for some tasks in NLP including language understanding tasks and thus the Transformer fails to generalize in those tasks (Dehghani et al., 2018). We also investigate the usage of the Universal Transformer for this task to see how it compares to the Transformer model.

We focus on self-attentional sequence modelling for this study and intend to provide an answer for one specific question which is:

- How effective are self-attentional models for training end-to-end task-oriented chatbots?

Our contribution in this study is as follows:

- We train end-to-end task-oriented chatbots using both self-attentional models and common recurrence-based models used in sequence modelling tasks and compare and analyze the results using different evaluation metrics on three different datasets.

- We provide insight into how effective are self-attentional models for this task and benchmark the time performance of these models against the recurrence-based sequence modelling methods.

- We try to quantify the effectiveness of self-attention mechanism in self-attentional models and compare its effect to recurrence-based models for the task of training end-to-end task-oriented chatbots.

2 Related Work

2.1 Task-Oriented Chatbots Architectures

End-to-end architectures are among the most used architectures for research in the field of conversational AI. The advantage of using an end-to-end architecture is that one does not need to explicitly train different components for language understanding and dialogue management and then concatenate them together. Network-based end-to-end task-oriented chatbots as in (Wen et al., 2016; Bordes et al., 2016) try to model the learning task as a policy learning method in which the model learns to output a proper response given the current state of the dialogue. As discussed before, all encoder-decoder sequence modelling methods can be used for training end-to-end chatbots. Eric and Manning (2017a) use the copy mechanism augmentation on simple recurrent neural sequence modelling and achieve good results in training end-to-end task-oriented chatbots (Gu et al., 2016).

Another popular method for training chatbots is based on memory networks. Memory networks augment the neural networks with task-specific memories which the model can learn to read and write. Memory networks have been used in (Bordes et al., 2016) for training task-oriented agents in which they store dialogue context in the memory module, and then the model uses it to select a system response (also stored in the memory module) from a set of candidates. A variation of Key-value memory networks (Miller et al., 2016) has been used in (Eric and Manning, 2017b) for the training task-oriented chatbots which stores the knowledge base in the form of triplets (which is (subject,relation,object) such as (yoga,time,3pm)) in the key-value memory network and then the model tries to select the most relevant entity from the memory and create a relevant response. This approach makes the interaction with the knowledge base smoother compared to other models.

Another approach for training end-to-end task-oriented dialogue systems tries to model the task-oriented dialogue generation in a reinforcement learning approach in which the current state of the conversation is passed to some sequence learning network, and this network decides the action which the chatbot should act upon. End-to-end LSTM based model (Williams and Zweig, 2016), and the Hybrid Code Networks (Williams et al., 2017) can use both supervised and reinforcement learning approaches for training task-oriented chatbots.

2.2 Sequence Modelling Methods

Sequence modelling methods usually fall into recurrence-based, convolution-based, and self-attentional-based methods. In recurrence-based sequence modeling, the words are fed into the model in a sequential way, and the model learns the dependencies between the tokens given the context from the past (and the future in case of bidirectional Recurrent Neural Networks (RNNs)) (Goodfellow et al., 2016). RNNs and their variations such as Long Short-term Memory

(LSTM) (Hochreiter and Schmidhuber, 1997), and Gated Recurrent Units (GRU) (Cho et al., 2014) are the most widely used recurrence-based models used in sequence modelling tasks. Convolution-based sequence modelling methods rely on Convolutional Neural Networks (CNN) (LeCun et al., 1998) which are mostly used for vision tasks but can also be used for handling sequential data. In CNN-based sequence modelling, multiple CNN layers are stacked on top of each other to give the model the ability to learn long-range dependencies. The stacking of layers in CNNs for sequence modeling allows the model to grow its receptive field, or in other words context size, and thus can model complex dependencies between different sections of the input sequence (Gehring et al., 2017; Yu and Koltun, 2015). WaveNet (2016), used in audio synthesis, and ByteNet (2016), used in machine translation tasks, are examples of models trained using convolution-based sequence modelling.

3 Models

We compare the most commonly used recurrence-based models for sequence modelling and contrast them with Transformer and Universal Transformer models. The models that we train are:

3.1 LSTM and Bi-Directional LSTM

Long Short-term Memory (LSTM) networks are a special kind of RNN networks which can learn long-term dependencies (Hochreiter and Schmidhuber, 1997). RNN models suffer from the vanishing gradient problem (Bengio et al., 1994) which makes it hard for RNN models to learn long-term dependencies. The LSTM model tackles this problem by defining a gating mechanism which introduces input, output and forget gates, and the model has the ability to decide how much of the previous information it needs to keep and how much of the new information it needs to integrate and thus this mechanism helps the model keep track of long-term dependencies.

Bi-directional LSTMs (Schuster and Paliwal, 1997) are a variation of LSTMs which proved to give better results for some NLP tasks (Graves and Schmidhuber, 2005). The idea behind a Bi-directional LSTM is to give the network (while training) the ability to not only look at past tokens, like LSTM does, but to future tokens, so the model has access to information both form the past and future. In the case of a task-oriented dialogue generation systems, in some cases, the information needed so that the model learns the dependencies between the tokens, comes from the tokens that are ahead of the current index, and if the model is able to take future tokens into accounts it can learn more efficiently.

3.2 Transformer

As discussed before, Transformer is the first model that entirely relies on the self-attention mechanism for both the encoder and the decoder. The Transformer uses the self-attention mechanism to learn a representation of a sentence by relating different positions of that sentence. Like many of the sequence modelling methods, Transformer follows the encoder-decoder architecture in which the input is given to the encoder and the results of the encoder is passed to the decoder to create the output sequence. The difference between Transformer (which is a self-attentional model) and other sequence models (such as recurrence-based and convolution-based) is that the encoder and decoder architecture is only based on the self-attention mechanism. The Transformer also uses multi-head attention which intends to give the model the ability to look at different representations of the different positions of both the input (encoder self-attention), output (decoder self-attention) and also between input and output (encoder-decoder attention) (Vaswani et al., 2017). It has been used in a variety of NLP tasks such as mathematical language understanding [110], language modeling (Dai et al., 2018), machine translation (Vaswani et al., 2017), question answering (Devlin et al., 2018), and text summarization (Liu et al., 2018).

3.3 Universal Transformer

The Universal Transformer model is an encoder-decoder-based sequence-to-sequence model which applies recurrence to the representation of each of the positions of the input and output sequences. The main difference between the RNN recurrence and the Universal Transformer recurrence is that the recurrence used in the Universal Transformer is applied on consecutive representation vectors of each token in the sequence (i.e., over depth) whereas in the RNN models this recurrence is applied on positions of the tokens in the sequence. A variation of the Universal Transformer, called Adaptive Universal

Transformer, applies the Adaptive Computation Time (ACT) (Graves, 2013) technique on the Universal Transformer model which makes the model train faster since it saves computation time and also in some cases can increase the model accuracy. The ACT allows the Universal Transformer model to use different recurrence time steps for different tokens.

We know, based on reported evidence that transformers are potent in NLP tasks like translation and question answering. Our aim is to assess the applicability and effectiveness of transformers and universal-transformers in the domain of task-oriented conversational agents. In the next section, we report on experiments to investigate the usage of self-attentional models performance against the aforementioned models for the task of training end-to-end task-oriented chatbots.

4 Experiments

We run our experiments on Tesla 960M Graphical Processing Unit (GPU). We evaluated the models using the aforementioned metrics and also applied early stopping (with delta set to 0.1 for 600 training steps).

4.1 Datasets

We use three different datasets for training the models. We use the Dialogue State Tracking Competition 2 (DSTC2) dataset (Williams et al., 2013) which is the most widely used dataset for research on task-oriented chatbots. We also used two other datasets recently open-sourced by Google Research (Shah et al., 2018) which are M2M-sim-M (dataset in movie domain) and M2M-sim-R (dataset in restaurant domain)[1]. M2M stands for Machines Talking to Machines which refers to the framework with which these two datasets were created. In this framework, dialogues are created via dialogue self-play and later augmented via crowdsourcing. We trained on our models on different datasets in order to make sure the results are not corpus-biased. Table 1 shows the statistics of these three datasets which we will use to train and evaluate the models.

The M2M dataset has more diversity in both language and dialogue flow compared to the the commonly used DSTC2 dataset which makes it appealing for the task of creating task-oriented chatbots. This is also the reason that we decided to use M2M dataset in our experiments to see how well models can handle a more diversed dataset.

Dataset	Num. of Slots	Train	Dev	Test
DSTC2	8	1618	1117	500
M2M-R	9	1116	349	775
M2M-M	5	384	120	264

Table 1: Statistics of DSTC2, M2M-R, and M2M-M Datasets

4.1.1 Dataset Preparation

We followed the data preparation process used for feeding the conversation history into the encoder-decoder as in (Eric and Manning, 2017a). Consider a sample dialogue D in the corpus which consists of a number of turns exchanged between the user and the system. D can be represented as $(u_1, s_1), (u_2, s_2), ..., (u_k, s_k)$ where k is the number of turns in this dialogue. At each time step in the conversation, we encode the conversation turns up to that time step, which is the context of the dialogue so far, and the system response after that time step will be used as the target. For example, given we are processing the conversation at time step i, the context of the conversation so far would be $(u_1, s_1, u_2, s_2, ..., u_i)$ and the model has to learn to output (s_i) as the target.

4.2 Training

We used the tensor2tensor library (Vaswani et al., 2018) in our experiments for training and evaluation of sequence modeling methods. We use Adam optimizer (Kingma and Ba, 2014) for training the models. We set $\beta_1 = 0.9$, $\beta_2 = 0.997$, and $\epsilon = 1e - 9$ for the Adam optimizer and started with learning rate of 0.2 with noam learning rate decay schema (Vaswani et al., 2017). In order to avoid overfitting, we use dropout (Srivastava et al., 2014) with dropout chosen from [0.7-0.9] range. We also conducted early stopping (Goodfellow et al., 2016) to avoid overfitting in our experiments as the regularization methods. We set the batch size to 4096, hidden size to 128, and the embedding size to 128 for all the models. We also used grid search for hyperparameter tuning for all of the trained models. Details of our training and hyperparameter tuning and the code for reproducing the results can be found in the *chatbot-exp*

[1]https://github.com/google-research-datasets/simulated-dialogue

Dataset Split	Model	BLEU	Per Turn. Acc	Per Diag. Acc	Entity F1
test	LSTM (bs=1)	5.75	17.70	0.0	5.63
	LSTM + Attention (bs=2)	30.84	18.08	0.15	32.16
	Bi-LSTM (bs=2)	30.38	18.04	0.0	24.34
	Bi-LSTM + Attention (bs=2)	38.64	26.04	0.62	43.52
	Transformer (bs=2)	**51.83**	**39.02**	**1.7**	**64.20**
	UT (bs=2)	44.93	36.62	1.08	57.98
	UT + ACT (bs=2)	39.40	30.00	0.15	61.49
development	LSTM	16.13	10.33	0.0	6.54
	LSTM + Attention	31.05	18.68	0.31	32.59
	Bi-LSTM	30.92	19.07	0.31	25.91
	Bi-LSTM + Attention	39.12	27.28	**0.96**	44.15
	Transformer	**54.18**	**41.09**	0.62	**66.02**
	UT	47.95	39.01	0.31	61.27
	UT + ACT	39.27	29.30	0.31	62.50

Table 2: Evaluation of Models on DSTC2 dataset for both test and development datasets (bs: shows the best beam size in inference; UT: Universal Transformers)

github repository[2].

4.3 Inference

In the inference time, there are mainly two methods for decoding which are greedy and beam search (Freitag and Al-Onaizan, 2017). Beam search has been proved to be an essential part in generative NLP task such as neural machine translation (Wu et al., 2016). In the case of dialogue generation systems, beam search could help alleviate the problem of having many possible valid outputs which do not match with the target but are valid and sensible outputs. Consider the case in which a task-oriented chatbot, trained for a restaurant reservation task, in response to the user utterance *"Persian food"*, generates the response *"what time and day would you like the reservation for?"* but the target defined for the system is *"would you like a fancy restaurant?"*. The response generated by the chatbot is a valid response which asks the user about other possible entities but does not match with the defined target.

We try to alleviate this problem in inference time by applying the beam search technique with a different beam size $\alpha \in \{1, 2, 4\}$ and pick the best result based on the BLEU score. Note that when $\alpha = 1$, we are using the original greedy search method for the generation task.

4.4 Evaluation Measures

BLEU: We use the Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002) metric which is commonly used in machine translation tasks. The BLEU metric can be used to evaluate dialogue generation models as in (Eric and Manning, 2017a; Li et al., 2015). The BLEU metric is a word-overlap metric which computes the co-occurrence of N-grams in the reference and the generated response and also applies the brevity penalty which tries to penalize far too short responses which are usually not desired in task-oriented chatbots. We compute the BLEU score using all generated responses of our systems.

Per-turn Accuracy: Per-turn accuracy measures the similarity of the system generated response versus the target response. Eric and Manning (2017a) used this metric to evaluate their systems in which they considered their response to be correct if all tokens in the system generated response matched the corresponding token in the target response. This metric is a little bit harsh, and the results may be low since all the tokens in the generated response have to be exactly in the same position as in the target response.

Per-Dialogue Accuracy: We calculate per-dialogue accuracy as used in (Bordes et al., 2016; Eric and Manning, 2017a). For this metric, we consider all the system generated responses and compare them to the target responses. A dialogue is considered to be true if all the turns in the system generated responses match the corresponding turns in the target responses. Note that this is a very strict metric in which all the utterances in the

[2]https://github.com/msaffarm/chatbot-exp

Dataset Split	Model	BLEU	Per Turn. Acc	Per Diag. Acc	Entity F1
M2M-R	LSTM(bs=2)	6.00	**2.3**	0.0	7.99
(test)	LSTM+Att.(bs=1)	7.9	1.84	0.0	16.77
	Bi-LSTM(bs=1)	8.15	1.8	0.0	19.61
	Bi-LSTM+Att.(bs=1)	8.3	0.97	0.0	24.12
	Transformer(bs=1)	**10.28**	1.76	0.0	**36.92**
	UT(bs=2)	9.15	1.88	0.0	25.44
	UT+ACT(bs=2)	8.54	1.43	0.0	23.12
M2M-M	LSTM(bs=4)	7.7	**3.36**	0.0	31.07
(test)	LSTM+Att.(bs=2)	8.3	3.27	0.0	31.18
	Bi-LSTM(bs=2)	9.6	2.09	0.0	28.09
	Bi-LSTM+Att.(bs=2)	10.62	2.54	0.0	32.43
	Transformer(bs=1)	**11.95**	2.36	0.0	**39.89**
	UT(bs=2)	10.87	3.15	0.0	34.15
	UT+ACT(bs=2)	10.48	2.46	0.0	32.76

Table 3: Evaluation of models on M2M restaurant (M2M-R) and movie (M2M-M) dataset for test datasets (bs: The best beam size in inference; UT: Universal Transformers)

dialogue should be the same as the target and in the right order.

F1-Entity Score: Datasets used in task-oriented chores have a set of entities which represent user preferences. For example, in the restaurant domain chatbots common entities are meal, restaurant name, date, time and the number of people (these are usually the required entities which are crucial for making reservations, but there could be optional entities such as location or rating). Each target response has a set of entities which the system asks or informs the user about. Our models have to be able to discern these specific entities and inject them into the generated response. To evaluate our models we could use named-entity recognition evaluation metrics (Jiang et al., 2016). The F1 score is the most commonly used metric used for the evaluation of named-entity recognition models which is the harmonic average of precision and recall of the model. We calculate this metric by micro-averaging over all the system generated responses.

5 Results and Discussion

5.1 Comparison of Models

The results of running the experiments for the aforementioned models is shown in Table 2 for the DSTC2 dataset and in Table 3 for the M2M datasets. The bold numbers show the best performing model in each of the evaluation metrics. As discussed before, for each model we use different beam sizes (bs) in inference time and report the best one. Our findings in Table 2 show that self-attentional models outperform common recurrence-based sequence modelling methods in the BLEU, Per-turn accuracy, and entity F1 score. The reduction in the evalution numbers for the M2M dataset and in our investigation of the trained model we found that this considerable reduction is due to the fact that the diversity of M2M dataset is considerably more compared to DSTC2 dataset while the traning corpus size is smaller.

5.2 Time Performance Comparison

Table 4 shows the time performance of the models trained on DSTC2 dataset. Note that in order to get a fair time performance comparison, we trained the models with the same batch size (4096) and on the same GPU. These numbers are for the best performing model (in terms of evaluation loss and selected using the early stopping method) for each of the sequence modelling methods. Time to Convergence (T2C) shows the approximate time that the model was trained to converge. We also show the loss in the development set for that specific checkpoint.

5.3 Effect of (Self-)Attention Mechanism

As discussed before in Section 3.2, self-attentional models rely on the self-attention mechanism for sequence modelling. Recurrence-based models such as LSTM and Bi-LSTM can also be augmented in order to increase their performance, as evident in Table 2 which shows the increase in the performance of both LSTM and Bi-LSTM

Model	T2C (sec)	Dev Loss
LSTM	1100	0.89
LSTM+Att	1305	0.62
Bi-LSTM	1865	0.60
Bi-LSTM+Att	2120	0.49
Transformer	**612**	**0.31**
UT	1939	0.36
UT+ACT	665	0.33

Table 4: Comparison of convergence performance of the models

when augmented with an attention mechanism. This leads to the question whether we can increase the performance of recurrence-based models by adding multiple attention heads, similar to the multi-head self-attention mechanism used in self-attentional models, and outperform the self-attentional models.

To investigate this question, we ran a number of experiments in which we added multiple attention heads on top of Bi-LSTM model and also tried a different number of self-attention heads in self-attentional models in order to compare their performance for this specific task. Table 6 shows the results of these experiments. Note that the models in Table 6 are actually the best models that we found in our experiments on DSTC2 dataset and we only changed one parameter for each of them, i.e. the number of attention heads in the recurrence-based models and the number of self-attention heads in the self-attentional models, keeping all other parameters unchanged. We also report the results of models with beam size of 2 in inference time. We increased the number of attention heads in the Bi-LSTM model up to 64 heads to see its performance change. Note that increasing the number of attention heads makes the training time intractable and time consuming while the model size would increase significantly as shown in Table 5. Furthermore, by observing the results of the Bi-LSTM+Att model in Table 6 (both test and development set) we can see that Bi-LSTM performance decreases and thus there is no need to increase the attention heads further.

Our findings in Table 6 show that the self-attention mechanism can outperform recurrence-based models even if the recurrence-based models have multiple attention heads. The Bi-LSTM model with 64 attention heads cannot beat the best Trasnformer model with NH=4 and also its results are very close to the Transformer model with NH=1. This observation clearly depicts the power of self-attentional based models and demonstrates that the attention mechanism used in self-attentional models as the backbone for learning, outperforms recurrence-based models even if they are augmented with multiple attention heads.

Model	T2C (sec)	Dev Loss
Bi-LSTM+Att.[NH=1]	2120	0.49
Bi-LSTM+Att.[NH=4]	3098	0.47
Bi-LSTM+Att.[NH=8]	3530	0.44
Bi-LSTM+Att.[NH=16]	3856	0.44
Bi-LSTM+Att.[NH=32]	7320	0.36
Bi-LSTM+Att.[NH=64]	**9874**	**0.38**
Transformer[NH=1]	**375**	**0.33**
Transformer[NH=4]	**612**	**0.31**
Transformer[NH=8]	476	0.31

Table 5: Comparison of convergence performance of the models

6 Conclusion and Future Work

We have determined that Transformers and Universal-Transformers are indeed effective at generating appropriate responses in task-oriented chatbot systems. In actuality, their performance is even better than the typically used deep learning architectures. Our findings in Table 2 show that self-attentional models outperform common recurrence-based sequence modelling methods in the BLEU, Per-turn accuracy, and entity F1 score. The results of the Transformer model beats all other models in all of the evaluation metrics. Also, comparing the result of LSTM and LSTM with attention mechanism as well as the Bi-LSTM with Bi-LSTM with attention mechanism, it can be observed in the results that adding the attention mechanism can increase the performance of the models. Comparing the results of self-attentional models shows that the Transformer model outperforms the other self-attentional models, while the Universal Transformer model gives reasonably good results.

In future work, it would be interesting to compare the performance of self-attentional models (specifically the winning Transformer model) against other end-to-end architectures such as the Memory Augmented Networks.

Dataset Split	Model	BLEU	Per-Turn Acc	Per-Diag Acc	Entity F1
test	Bi-LSTM+Att.[NH=1]	38.64	26.04	0.62	43.52
	Bi-LSTM+Att.[NH=4]	42.23	29.01	0.92	48.06
	Bi-LSTM+Att.[NH=8]	42.61	28.18	0.77	49.90
	Bi-LSTM+Att.[NH=16]	**43.11**	**30.34**	**0.61**	**50.87**
	Bi-LSTM+Att.[NH=32]	**48.62**	**36.46**	**1.85**	**59.8**
	Bi-LSTM+Att.[NH=64]	**47.33**	**33.17**	**1.23**	**56.49**
	Transformer[NH=1]	**45.90**	**36.64**	**1.7**	**57.55**
	Transformer[NH=4]	**51.83**	**39.02**	**1.7**	**64.20**
	Transformer[NH=8]	51.37	39.45	3.24	62.38
	UT[NH=1]	43.02	31.20	1.54	60.10
	UT[NH=8]	48.17	35.76	2.93	61.56
	UT+ACT[NH=1]	34.98	25.66	0.46	51.32
	UT+ACT[NH=8]	36.29	24.97	0.31	55.27
development	Bi-LSTM+Att.[NH=1]	39.12	27.28	0.96	44.15
	Bi-LSTM+Att.[NH=4]	40.47	27.64	0.93	48.10
	Bi-LSTM+Att. [NH=8]	42.78	28.36	0.31	50.05
	Bi-LSTM+Att.[NH=16]	**42.88**	**30.36**	**0.93**	**52.09**
	Bi-LSTM+Att.[NH=32]	**49.36**	**38.24**	**0.61**	**61.26**
	Bi-LSTM+Att.[NH=64]	**47.28**	**33.12**	**0.93**	**56.86**
	Transformer[NH=1]	**47.86**	**38.33**	**1.85**	**60.37**
	Transformer[NH=4]	**54.18**	**41.09**	**0.62**	**66.02**
	Transformer[NH=8]	51.54	39.42	1.54	63.56
	UT[NH=1]	43.01	32.12	1.58	60.42
	UT[NH=8]	47.89	35.57	1.23	61.33
	UT+ACT[NH=1]	35.74	26.46	0.31	52.71
	UT+ACT[NH=8]	38.95	27.10	0.31	57.02

Table 6: Evaluation of effect of self-attention mechanism using DSTC2 dataset (Att: Attetnion mechanism; UT: Universal Transformers; ACT: Adaptive Computation Time; NH: Number of attention heads)

References

Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5(2):157–166.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683* .

Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter* 19(2):25–35.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2018. Transformer-xl: Language modeling with longer-term dependency .

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. 2018. Universal transformers. *arXiv preprint arXiv:1807.03819* .

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2015. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931* .

Mihail Eric and Christopher D Manning. 2017a. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. *arXiv preprint arXiv:1701.04024* .

Mihail Eric and Christopher D Manning. 2017b. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414* .

Markus Freitag and Yaser Al-Onaizan. 2017. Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806* .

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, pages 1243–1252.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks* 18(5-6):602–610.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393* .

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

Ridong Jiang, Rafael E Banchs, and Haizhou Li. 2016. Evaluating and combining name entity recognition systems. In *Proceedings of the Sixth Named Entity Workshop*. pages 21–27.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* .

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198* .

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126* .

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, pages 311–318.

Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, et al. 2018. Conversational ai: The science behind the alexa prize. *arXiv preprint arXiv:1801.03604* .

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871* .

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364* .

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. *SSW* 125.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, et al. 2018. Tensor2tensor for neural machine translation. *arXiv preprint arXiv:1803.07416* .

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

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562* .

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*. pages 404–413.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274* .

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269* .

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122* .