CS230

ServiceNet: End-to-End Dialog System for Technical Support

Madhulima Pandey
mpandey8@stanford.edu

David Uvalle
davidu@stanford.edu

Abstract

We investigate an end-to-end goal-oriented dialog system for purpose-specific domains. We have focused on the problem of automating IT tech support. We explore various end-to-end learning solution architectures. We present evaluation metrics and conclude that Bidirectional Long Short-Term Memory (LSTM) and pre-trained Bidirectional Encoder Representation from Transformer (BERT) networks generate the best results on the Ubuntu Dialog Corpus (UDC) dataset. The transfer learning through pre-trained BERT is the most promising solution as it has better accuracy and runtime than Bidirectional LSTM.

1 Introduction

An end-to-end goal-oriented dialog system aims at providing accurate responses to user queries during a dialog. IT tech support is a manually intensive task and this domain contains rich support logs and hence provides a great opportunity for building dialog agents using deep learning methods. Traditional dialog systems have a complex pipeline consisting of independently developed components of spoken language understanding. Such a system requires a lot of handcrafted rules and domain specific knowledge. Furthermore, these complex pipelines are prone to cascading errors from each stage. In this paper, we use Ubuntu Dialog Corpus (UDC) [5], that consists of context-response pairs, to explore an end-to-end learning model that takes input context c, and outputs scores for candidate responses r and then selects k best responses as defined by the model parameters.

2 Related work

2.1 Task-Oriented Dialog Pipeline Systems

Figure 1 shows a typical pipeline architecture of a task-oriented dialog system. The dialog systems consists of a number of components connected together in a pipeline which make it hard to track source of errors and optimize the system for a global target [1].

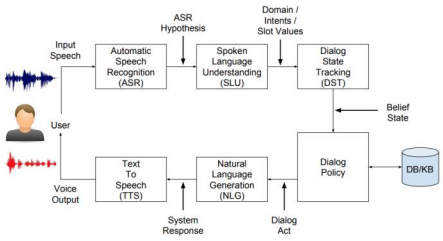

Figure 1: Pipeline architecture for dialog systems [1]

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

# dialogues	930,000
# utterances	7,100,000
# words total	100,000,000
Min # turns	3
Avg # turns	7.71
Avg # words	10.34
Vocab size	91,000

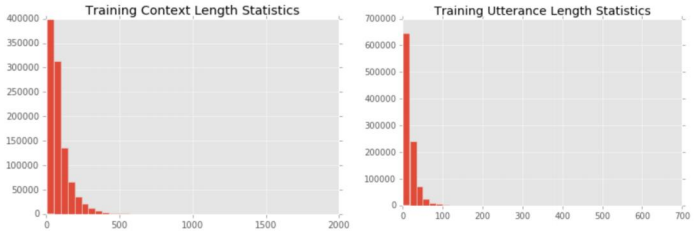

Figure 2: UDC Dataset Size and Characteristics

2.2 End-to-end Dialog Models

Wen et. al [2] proposed end-to-end dialog model with connected systems components such as Speech language Understanding(SLU), Dialog State Tracking(DST), and dialog policy. This model suffers from the limitation of components being trained separately and does not generalize well to unseen dialogue states. Bordes et. al [3] proposed a mode that uses an RNN to encode the dialog state and used end-to-end memory network[weston]. We have used similar approach to generate encodings for the the dialog using LSTM. Williams et.al [4] have proposed a hybrid network for dialog that can be trained with supervised and reinforcement learning. We plan to extend our work to use reinforcement learning to supplement the current supervised learning models.

3 Dataset and Features

The Ubuntu Dialogue Corpus [5] is created from a collection of logs from Ubuntu related chat rooms on the IRC network. These chatrooms are commonly used for obtaining technical support with various Ubuntu issues. Users ask a general question about a problem they are having with Ubuntu. Then, another more experienced user replies with a potential solution, after first addressing the username of the first user. Basic statistics on the dataset is presented in Figure 2. This data possesses the following characteristics: i. Two-way conversations between humans, ii. Large number of conversations, iii. Multi-turn conversations (more than 3), and iv. Task-specific conversation as opposed to general purpose.

3.1 Data preprocessing

We process the raw data into training data that contains context, response utterance which we aim to correctly identify, and a flag label that indicates whether or not the response was the actual next utterance after the given context. In the context __eou__ means end of utterance without a change of turn, and __eot__ means end of turn. Therefore the context represent the dialog between to chat users. This is illustrated in figures 3 and 4. The dataset csv files are converted into tfrecord files that contained Google's protobuf serialized representation of the data. To create this representation, each distinct word is converted into a unique integer from the dictionary of vocab words. In case of BERT, we had to change the conversion process, where the context and the responses were converted into BERT encodings and then converted into a tfrecord file.

	Context	Utterance	Label
0	i think we could import the old comment via rsync , but from there we need to go via email . i think it be easier than cach the status on each bug and than import bite here and there __eou__ __eot__ it would be veri easi to keep a hash db of message-id __eou__ sound good __eou__ __eot__ ok __eou__ perhap we can ship an ad-hoc apt_preferec __eou__ __eot__ version ? __eou__ __eot__ thank __eou__ __eot__ not yet __eou__ it be cover by your insur ? __eou__ __eot__ yes __eou__ but it 's realli no...	basic each xfree86 upload will not forc user to upgrad 100mb of font for noth __eou__ no someth i do in my spare time . __eou__	1
1	i 'm not suggest all - onli the one you modifi . __eou__ __eot__ ok , it sound like you re agre with me , then __eou__ though rather than " the one we modifi '' , my idea be " the one we need to merg '' __eou__ __eot__	sorri __eou__ i think it be ubuntu relat . __eou__	0

Figure 3: UDC Training Set

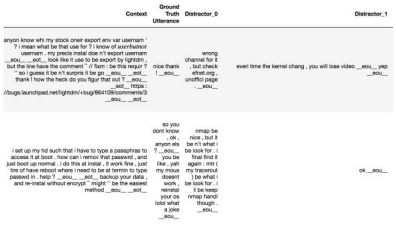

Figure 4: UDC Test Set

The validation and test sets contain context, ground truth, and multiple distractors. In the Figure 4 we show two distractors with one ground truth for a given context. Having multiple distractors is useful when computing Recall@k metrics as these can be used to alter the difficulty level of the classification task. We have used train, validation and test split of 96%, 2%, and 2%.

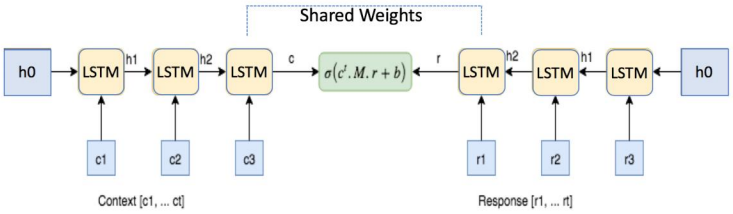

Figure 5: Dual encoder model.

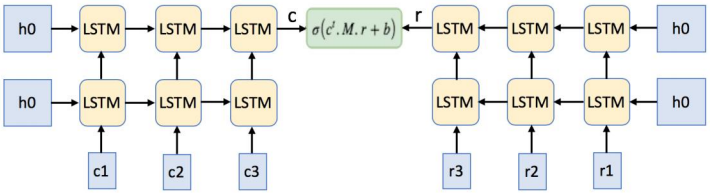

Figure 6: Multi-layer LSTM Network.

4 Methods

We have employed three approaches to evaluate the performance of end-to-end dialog systems on UDC dataset: classical information retrieval methods, supervised learning models with LSTMs and transfer learning with pre-trained models using BERT [6].

4.1 TF-IDF

We used TF-IDF, commonly used in document classification, as the baseline model. TF-IDF captures how important a given word is in the context. TF-IDF vectors are computed for the context and all the candidate response, and the one with the highest cosine similarity is selected as the correct response [5].

4.2 LSTM models

4.2.1 Single layer LSTM

LSTM units capture longer-term dependencies in sequences. We implemented a Siamese network with shared weights to produce the encodings of the context and the response , similar to Lowe et al. [5]. This *Dual Encoder* architecture is shown in Figure 5. The words [$c1$, $c2$, ...] in the $context$ and words [$r1$, $r2$, ...] in the $response$ are each converted into embeddings using GloVe [8]. These embeddings are the inputs to the LSTM cells. The output of the last LSTM cell represents the encoding of the context and the response as indicated by c and r, respectively in the figure.

Assuming the the context and response embeddings, c, and r are of size d, similar to a generative approach, a new context c' is computed by multiplying a weight matrix $M \in R^{dxd}$ with the response embedding to $c' = Mr$. The inner product of this new context, c', is multiplied with the original context, c, and bias $b \in R^d$ is added to the product, to get distance measurement and this distance is used as an argument to a sigmoid function to calculate the probability of a valid pair [5]:

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

4.2.2 Multi-layer LSTM

Multi-layer LSTMs have additional hidden layers that capture the learned representations of prior layers and create new representations at a higher level of abstraction. In the multi-layer model, the hidden state of the top layer at the end of the sequence generates the context and response encoding (Figure 6). The the loss function is same as in single-layer LSTM above.

4.2.3 Bi-directional LSTM network

Bi-directional LSTM networks duplicate the first recurrent layer in the network so that there are two layers side-by-side (Figure 7). These networks have access to the past and future information which

allows the context of the whole utterance to generate the encoding. The encoding is constructed by concatenating the hidden cell states from the forward pass and the backward pass networks, respectively, as illustrated in the figure.

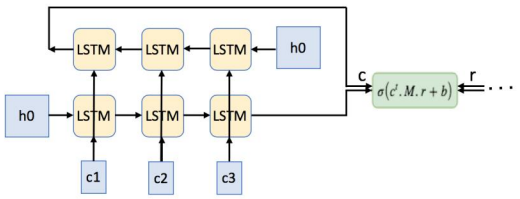

Figure 7: Bidirectional LSTM Network.

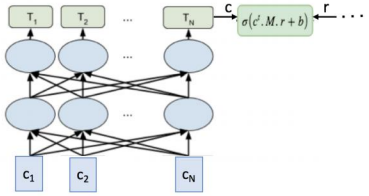

Figure 8: BERT-based dual encoder network.

4.2.4 Loss function

All the three variations of LSTM model are trained by minimizing cross entropy loss for all context c_i and response r_i pairs in the training dataset:

$$Loss = -\sum_{n} log(p(flag_n|c_n, r_n, M)) \tag{2}$$

4.3 Pre-trained BERT Model

BERT is a recent model that was published by Google AI group[6]. BERT is a multi-layer bi-directional transformer encoder. The Transformer applies a self-attention mechanism which directly models relationship of all words in a sentence regardless of their respective position. Pre-trained BERT model can be fine tuned to create state-of-the-art models for a variety of NLP tasks. In our project, we use a pre-trained BERT model to generate encodings of context c and responses r. We train the network to only learn the parameters of matrix M and bias b of the last layer as described in section 4.2.1. We used a pre-trained model with 12 Transformer Blocks and 768 hidden states, with 110M variables.[9]

4.4 Evaluation Metrics

We use the Recall@K metric for evaluating the different models mentioned above. The task is for the agent to select the k most likely responses, and it is correct if the true response is among the k candidates. In our case, we measure the Recall@K metric for a set of 10 responses, where one response is the correct one, and the remaining nine are distractors. Note that during training, we predict the probability of the example being correct. During evaluation, the goal is to score the response and the nine distractors and pick the k top ones.

5 Experiments

5.1 Hyperparameters

We experimented with several configurations of the models, including single layer (state sizes from 256 to 768), multi-layer LSTMs (2-3 layers - 128 state size), and bidirectional models. We finally employed the 12-transformer, 768 hidden size BERT pre-trained model as this model has presented the state-of-the-art results in a wide variety of NLP tasks.

We used GloVe word embedding size of 100. Additionally, we used a maximum sequence length of 160 for the recurrent models and BERT, i.e., the context and responses are truncated to 160 words. This was chosen based on the results from the paper[5].

We used a learning rate of 0.001, with Adam which gave the best results. The batch size of 64 was dictated by GPU memory limits (GTX 970). We discovered that after 30-40K epochs, the loss value minimized, and the highest Recall@K accuracy numbers were achieved.

We also experimented with using GloVe pre-trained embeddings for words in LSTM-based networks, and ran the same experiment without the GloVe word embeddings.

Network Type	State Size	# Layers	Bidirectional	Pre-Trained	Recall@1	Recall@2	Recall@5
LSTM	256	1	No	No	0.526	0.708	0.923
LSTM	256	1	No	Yes	0.519	0.708	0.919
LSTM	768	1	No	Yes	0.499	0.683	0.914
LSTM-Multi	128	2	No	Yes	0.494	0.689	0.911
LSTM-Multi	128	3	No	Yes	0.483	0.675	0.911
LSTM-Bidi	128	1	Yes	Yes	0.512	0.696	0.915
LSTM-Bidi	128	1	Yes	No	0.537	0.712	0.920
BERT	768	12	Yes	Yes	0.551	0.738	0.927

Table 1: Results - Recall@K scores for various network types

5.2 BERT Model

For BERT, we downloaded a pre-trained model with 12 Transformer Blocks and 768 Hidden states [9], and set up a sentence to embedding service with *BERT serving server* [10], to translate each of the context and responses into encodings ($\in R^{768}$), as shown in Figure 8. Note that since we are using pre-trained BERT models, this conversion to encoding needs to be done once for the entire dataset.

6 Results and Discussion

We have summarized the results of our runs in Table 1. Each of these networks gives a substantially better accuracy than the TD-IDF baseline values of Recall@5 of 77%, and Recall@1 of 41%, showing the advantages of recurrent models.

Our initial experiments focused on increasing the size of the LSTM cells, but state size beyond 256 did not result in improved performance. In fact, the performance at state size 768 was worse for all Recall@K metrics, likely due to overfitting.

Multi-layer LSTM topologies with 2 or even 3 layers did not result in improved performance over single layer LSTMs. However, we found that bi-directional LSTMs offered better performance than both single-layer and multi-layer LSTM models. This is likely due to the fact that bidirectional LSTMs can better capture the semantics of the utterances by evaluating both left and right contexts.

We experimented with the LSTM networks above with and without pre-trained GloVe embeddings for the words in the context and the responses. We did not find significant performance difference with the use of pre-trained embeddings.

Our experiments with Bidirectional Encoder Representations from Transformers (BERT) has yielded the best results. This is possibly because of the bidirectional and attention capabilities of the networks, with the joint conditioning on both left and right context in all layers. Note that BERT does away with LSTMs/GRUs completely. With this, the model has to only learn $M \in R^{768 \times 768}$, and $b \in R^{768}$ parameters. The encoding generation for training, validation and test csv dataset files takes over 7 hours. However, with the encoded values as input to the last layer of the network, the training runs about 20 times faster than the previous models with LSTM.

7 Conclusion/Future Work

Our conclusion is that end-to-end neural network based dialog systems for goal-oriented purpose-specific domains work well with recurrent network topologies, and we discovered that transfer learning through BERT yielded the best answers, over multi-layer and bidirectional LSTM architectures. Our experiments suggest that context is best captured by bidirectional topologies, as shown by the superior performance of BERT and bidirectional LSTM over uni-directional LSTM topologies, including multi-layer variations. BERT's advantage lies in its bidirectional architecture with self-attention mechanism which models relationship of all words in a sentence irrespective of their position.

In future work, we would like to add knowledge sources related to the goal-oriented task and additional context information to improve the accuracy of the next utterance selection. We would also like to extend our models with reinforcement learning to include user feedback such as correction of response and satisfaction scores.

8 Contributions and Code

Madhulima worked on the Tensorflow LSTM, multi-layer LSTM, bidirectional LSTM and BERT models. David worked on TF-IDF, RNN and GRU models in Keras. Madhulima and David contributed equally to the final report.

Our code is available at https://github.com/madhulima/dialog-system-project. The readme files contain instructions to run the code. The base code we worked off was from an implementation of basic model by Britz [12]. We modified the code to fix obselete TensorFlow dependencies and broken dataset dependencies. We implemented various LSTM models on top of this base code including multi-layer LSTM network and bi-directional LSTM network. We implemented code to generate BERT context and response encodings and integrated the BERT model with the end-to-end dialog system.

References

[1] Liu, B., & Lane, I. (2018). End-to-End Learning of Task-Oriented Dialogs. NAACL HLT 2018, 67.

[2] Wen, T. H., Vandyke, D., Mrksic, N., Gasic, M., Barahona, L. M. R., Su, P. H., ... & Young, S. (2017). A Network-based End-to-End Trainable Task-oriented Dialogue System. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers (Vol. 1, pp. 438-449).

[3] Bordes, A., Boureau, Y. L., & Weston, J. (2016). Learning end-to-end goal-oriented dialog. arXiv preprint arXiv:1605.07683.

[4] Williams, J. D., Asadi, K., & Zweig, G. (2017). Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. arXiv preprint arXiv:1702.03274.

[5] 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.

[6] 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.

[7] Yu, L., Hermann, K. M., Blunsom, P., & Pulman, S. (2014). Deep learning for answer sentence selection. arXiv preprint arXiv:1412.1632.

[8] Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[9] Bert pre-trained model (12-layer, 768-hidden, 12-heads, 110M parameters) `https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip`

[10] Bert Serving Server `https://pypi.org/project/bert-serving-server/`

[11] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016, November). Tensorflow: a system for large-scale machine learning. In OSDI (Vol. 16, pp. 265-283).

[12] Britz, D., Retrieval-Based Conversational Model in Tensorflow - https://github.com/dennybritz/chatbot-retrieval

[13] 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.