# End to End Dialog System for Telugu

**Prathyusha Danda**[1] **Prathyusha Jwalapuram**[2] **Manish Shrivastava**[3]
Language Technologies Research Center
Kohli Center on Intelligent Systems
International Institute of Information Technology
Hyderabad, India
[1]danda.prathyusha@research.iiit.ac.in
[2]prathyusha.jwalapuram@research.iiit.ac.in
[3]m.shrivastava@iiit.ac.in

## Abstract

This paper describes an end to end dialog system created using sequence to sequence learning and memory networks for Telugu, a low-resource language. We automatically generate dialog data for Telugu in the tourist domain, using a knowledge base that provides tourist place, type, tour time, etc. Using this data, we train a sequence to sequence model to learn system responses in the dialog. In order to add the query prediction for information retrieval (through API calls), we train a memory network. We also handle cases requiring updation of API calls and querying for additional information. Using the combination of sequence to sequence learning and memory network, we successfully create an end to end dialog system for Telugu.

## 1 Introduction

There have been few attempts to create dialog systems for Telugu, which are mostly rule-based systems using ad-hoc user interactions to test the system rather than over a set of prepared test dialogs (Sravanthi et al., 2015; Reddy and Bandyopadhyay, 2006). This is primarily due to a lack of dialog data as Telugu is a low-resource language. Wen et al. (2016) proclaim that the greatest bottleneck for statistical approaches to dialog system development is the collection of appropriate data which is especially true for task oriented dialog systems; that for task-oriented dialog systems, in-domain data is essential.

Dialog models using neural networks are able to leverage the large amounts of data to learn meaningful representations for natural language and generation strategies, and require only a minimal amount of domain knowledge and handcrafting (Serban et al., 2016). The neural networks are used to represent both dialog histories and to produce output either through a generative model that generates responses word-by-word conditioned on a dialog context (which is the model this paper uses) or through a discriminative model that is trained to select an appropriate response from a set of candidate responses (Serban et al., 2016). We use both the models for generating system responses in our dialog system.

Sequence to Sequence learning (Sutskever et al., 2014) has been used to build end-to-end trainable non-task-oriented conversational dialog systems (Vinyals and Le, 2015; Shang et al., 2015; Serban et al., 2015). This approach models dialog as a source to target sequence transduction problem, applying an encoder network (Cho et al., 2014) to encode a user query into a distributed vector representation of its semantics, which conditions a decoder network to generate each system response. This has been extended to a task-oriented system that interacts with a knowledge base by Wen et al. (2016).

End-to-end dialog systems are trained on past dialogs directly, with no assumptions made on the basis of the domain or on the structure of the dialog, which makes scaling up automatically to new domains easy (Bordes and Weston, 2017). As an end-to-end neural model, Memory Networks (Weston et al., 2015a), with an attention based architecture, showed promising results for non goal-oriented dialog (Dodge et al., 2016), and have also been applied to question answering (Weston et al., 2015b; Bordes et al., 2015) and language modelling (Sukhbaatar et al., 2015). However, goal-oriented dialog requires the system to ask questions to clearly define a user request, query knowledge bases, etc., as extended by Bordes and Weston (2017).

We first create a corpus of Telugu dialog data in the Tourist domain, which we then use to train our sequence to sequence and memory network models. We report our results for system response generation through the sequence to sequence model, and our results for API call generation, for retrieving information from knowledge base, through the memory network model. Through this combination of sequence to sequence learning and memory network, we successfully create an end-to-end dialog system for the tourist domain in Telugu.

After discussing Related Work in Section 2, we outline the tasks our system must perform in Section 3, then we discuss the motivation behind our system pipeline in Section 4, Section 5 describes dialog data creation strategy, followed by sequence-to-sequence model for producing system responses in Section 6, Section 7 deals with the memory network layer for generating API calls, finally followed by the conclusion and future work in Sections 8 and 9 respectively.

## 2 Related Work

Ritter et al. (2011) first proposed using generative probabilistic models to model conversations from micro-blogging websites, treating the response generation problem as a statistical machine translation problem, where the post is to be translated into a response. They find that generating responses is a harder problem than language translation due to the wide range of possible responses and a lack of alignment between the source and the response.

Shang et al. (2015) extend the work by using recurrent neural networks, which they show outperform retrieval-based and SMT based methods for generating responses to a post in Chinese, and are able to generate multiple responses with variety. Sordoni et al. (2015) go further by designing the response generation to be conditioned on past dialog utterances that provide contextual information, and also outperform MT and IR based models.

The end-to-end trainable, non-task-oriented conversational dialog systems built by Vinyals and Le (2015; Shang et al. (2015; Serban et al. (2015) using sequence to sequence learning (Sutskever et al., 2014) are promising chatbot systems but do not support domain specific tasks and do not interact with knowledge bases such as databases (Sukhbaatar et al., 2015; Yin et al., 2015), and therefore cannot provide useful information through their responses.

Wen et al. (2016) augment the sequence to sequence architecture with dialog history modelled by a set of belief trackers, and a distributed representation of user intent with delexicalisation and weight tying strategies. Their system provides relevant and appropriate responses at each turn and also interacts with a database through a slot-value pair representation of attributes. They achieve a high task success rate and show that the learned model can interact efficiently and naturally with human subjects to complete an application specific task.

Dodge et al. (2016) use Memory Networks (Weston et al., 2015a; Sukhbaatar et al., 2015) to train non goal oriented dialog, which showed promising results. Bordes and Weston (2017) train memory networks to perform tasks non-trivial tasks such as issuing API calls to knowledge bases and manipulating entities unseen in training; the bot is also able to ask questions to fill missing information. They show that memory networks can outperform a dedicated slot-filling rule-based baseline, and even classical IR and supervised embeddings; they solve the task of issuing API calls.

## 3 Tasks

As part of our dialog system, there are four main tasks we want to accomplish:

1. Generate appropriate system responses to user utterances.
2. Generate API calls for information retrieval
3. Generate API calls for information retrieval in case of updation.
4. Generate API calls for providing additional information

API calls represent specific operations that applications can invoke at runtime to perform tasks, one of which is querying data from the knowledge base[1]. In our implementation, the API calls simply consist of the keywords that we use to query the knowledge base to retrieve the highest rated tourist location or to give additional information such as address or phone number or opening time of a tourist location. Additional information (Task 4) consists of phone number, address and opening times.

[1]https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/calls.htm

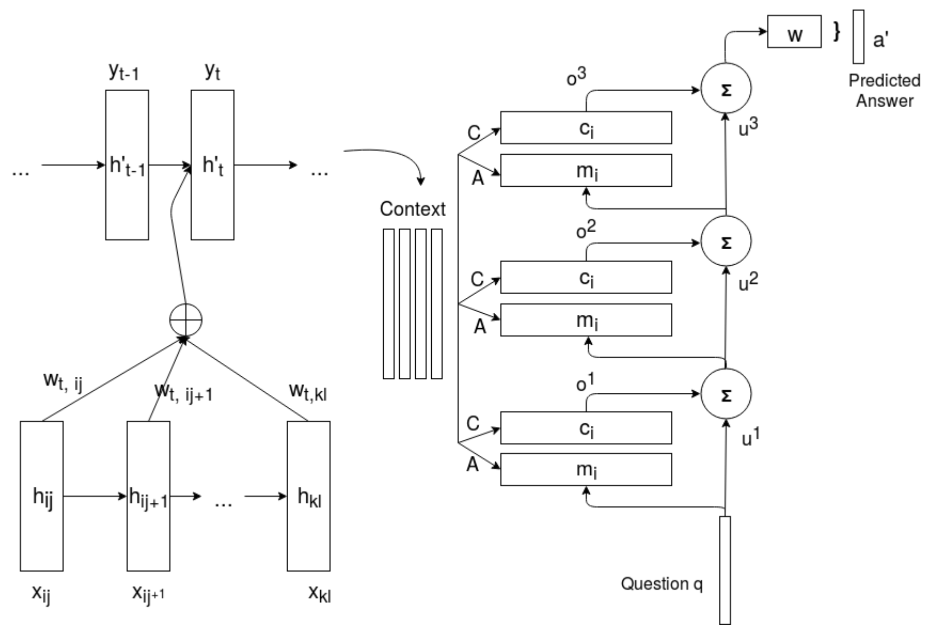


Figure 1: Network Architecture

## 4 Motivation behind System Pipeline

A sequence to sequence model generates a sentence by generating a sequence of words whereas in memory networks a system response is generated by picking one from all the possible dialog candidates, mentioned in Bordes and Weston (2017).

Hence, predicting system responses using a sequence to sequence model is preferable over memory networks. (See Figure 2)

## 5 Data Creation

In order to automatically create the substantial amount of data required to train an end-to-end dialog system, we use an approach similar to the one used by Bordes and Weston (2017). In Bordes and Weston (2017), data is simulated based on an underlying restaurant domain knowledge base that has attributes such as type of cuisine, location, etc. and can be queried using API calls.

Our knowledge base is similarly built on the tourist domain, and has the attributes area, type, tour duration, opening time, rating, phone number and address[2], of which area, type and tour duration are the 3 required keywords while opening time, phone number and address are query-able fields. Areas consist of place-names (such as Ban-

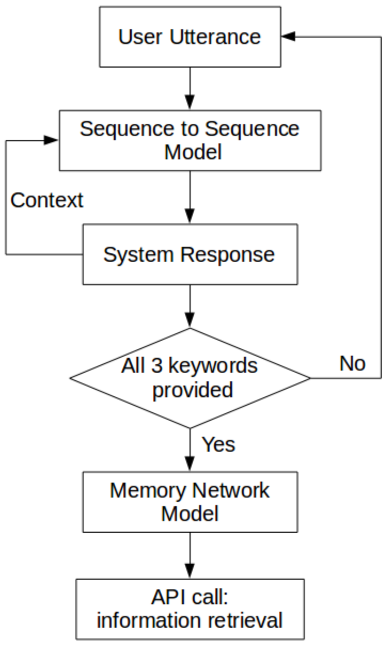


Figure 2: End to End System Pipeline

[2]https://en.wikipedia.org/wiki/List_of_tourist_attractions_in_Hyderabad

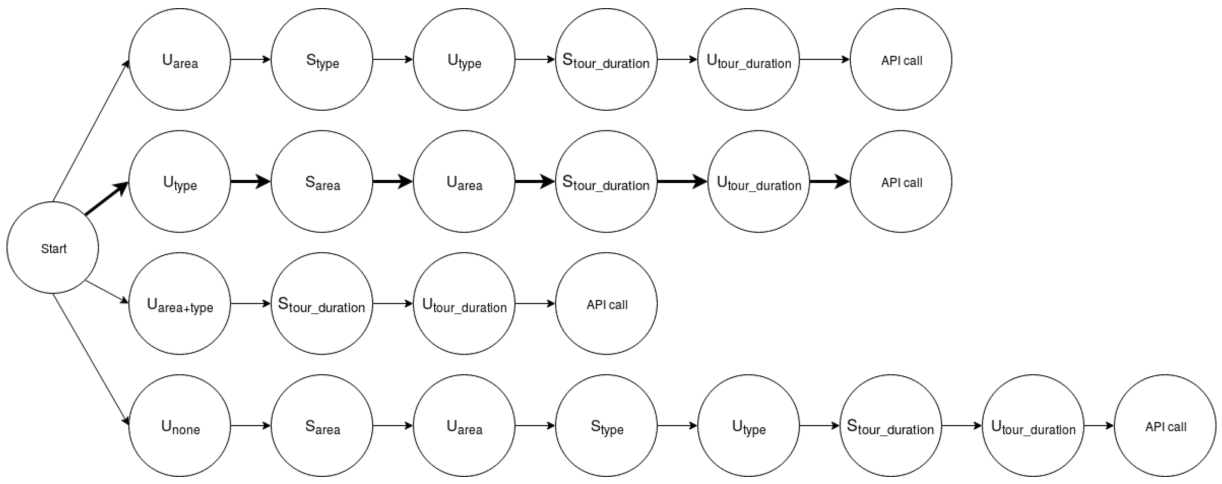


Figure 3: System Dialog Flow

jara Hills, Chanda Nagar Village, etc.) and types consist of types of tourist locations such as historical, religious, zoo and amusement park. The tour duration is the time in which the user wants to see a place, such as 15 minutes, 30 minutes, etc. There are a total of 30 areas, 4 types and 4 tour durations.

Based on this knowledge base, we generate queries by choosing any of the three required fields for a query, which are area, type and tour duration. Using natural language patterns in Telugu, we create user and system utterances. We append the keywords required for a query to the knowledge base at the end of each dialog in the form of an API call (e.g. api_call banjara hills zoo 15, api_call banjara hills zoo 15 address). However, since the knowledge base is in English, the API calls are also in English.

There are 68 possible patterns for the user to express 12 different intents and 9 possible patterns for the bot and a total of 1920 possible API calls. Although all the words in the language patterns are in Telugu, certain words which are commonly used in English are also transcribed and included, helping us handle some code-mixed cases.

Different permutations of the patterns combined with different entities in the knowledge base produce thousands of dialogs, which also include the API call that is required for information retrieval. See Figure 4 for an example of the dialog created for Tasks 1 and 2.

The user could begin by providing no information ($U_{none}$), which should prompt the system to ask for the area ($S_{area}$); the user could provide the area in the first utterance ($U_{area}$), which should prompt the system to ask for the type ($S_{type}$); the user may provide both the area and the type in the first utterance itself ($U_{place+type}$), which should prompt the system to ask for the tour duration ($S_{tour\_duration}$), and so on, and finally the system must be able to generate the correct corresponding API call to retrieve an appropriate tourist location or to give address or phone number of a particular tourist place (Task 1, 2 and 4).

The user can always request a change in the required fields, which should lead to an update in the API call. This forms a separate dataset for Task 3.

The possible paths are given in an example diagram Figure 3; the path taken in the example dialog in Figure 4 is highlighted.

The order of the fields can be jumbled through the utterances, but the system queries unfilled slots deterministically, i.e., the user may begin by specifying the type of location they want to visit and then be prompted to provide the area, whereas system will prompt for area, type and tour time in that order. Through this, we have 4 variations of the dialog beginnings: user specifying area first, user specifying type first, user specifying both area and type, and user specifying neither.

We partition the data differently (therefore producing differing amounts) for training the sequence to sequence model and the memory networks; this is described in their respective sections. The length of the dialogs varies from 5 to 12 system-user utterance pairs depending on the combinations. The maximum possible length of an utterance is 14.

We have released the data for public use so that those who wish to use the Telugu data for research

| User Utterance | System Response |
|---|---|
| నేను హిస్టారికల్ ప్లేస్ చూడాలని అనుకుంటున్నాను<br><br>*nEnu histArikal plEs cUdAlani anukuMtunnAnu*<br><br>I want to visit a historical place. | సరే. ఎక్కడ ఉండాలి<br><br>*sarE. ekkaduMdAli*<br><br>Okay. Where should it be? |
| నేను కాంచన  బాగ్ లో ఉన్న ప్రాంతాలు చూడాలని అనుకుంటున్నాను<br><br>*nEnu kAMcana bAg lO unnA prAMwAlu cUdAlani anukuMtunnanu*<br><br>I want to visit the places in Kanchanbagh. | సరే. ఎంత సమయం లో చూడాలి అనుకుంటున్నారు<br><br>*sarE. eMwa samayaM lO cUdAli anukuMtunnAru*<br><br>Okay. How long do you want to spend visiting? |
| 15 నిమిషాలలో చూడాలి అనుకుంటున్నాను<br><br>*15 nimiRAlalO cUdAli anukuMTunnAnu*<br><br>I want to spend 15 minutes. | సరే ఏ ప్రదేశాలు ఉన్నాయో చూసి చెప్తాను ఆగండి<br><br>*sarE. E praxESAlu unnAyO cUsi cepwAnu AgaMdi*<br><br>Okay. I will check which places are available and let you know.<br><br>api_call kanchanbagh historical 15 |

Figure 4: Example Dialog Created

can do so.[3]

## 6 System Response Generation Using Sequence to Sequence Learning

### 6.1 Training and Testing Data

In order to train a sequence to sequence learning model that can incorporate dialog history, the dialog data is partitioned in such a way as to provide instances with differing amounts of context. This means that to predict system response $S_t$, the input with context is $U_1, S_1, ..., S_{t-1}, U_t$. See Figure 5 for an example.

In each case, the queries are ordered deterministically; system must recognize the missing information and ask the user to provide it accordingly.

Through such partitioning, we obtain 180,000 instances of dialog with context. We sample 20,000 instances for training and separate 3,000 instances for testing for Task 1 and Task 2, and the same number of instances for Task 3 from its own dataset.

### 6.2 Architecture

We use an Encoder-Decoder (Sutskever et al., 2014) architecture for sequence to sequence learning, that uses one GRU (Cho et al., 2014) layer to encode the input sentence one timestep at a time to obtain a large fixed-dimensional vector representation, and then uses another GRU with attention to decode (Bahdanau et al., 2014) the output sequence for that vector.

In Figure 1, $x_{ij}$ corresponds to the one-hot vector of $j^{th}$ word in $i^{th}$ sequence. The $h_{ij}$ is a vector embedding $x_{ij}$. The $w_{t,ij}$ is the attention weight of $t^{th}$ word in the output sequence corresponding to $h_{ij}$. $y_t$ is the $t^{th}$ target word in the output sequence, with $h'_{t-1}$ being the RNN hidden state.

The embedding layer over the encoder takes a $|V_{in}|$ sized vector and outputs a vector of hidden size, where $|V_{in}|$ is the size of the input vocabulary and the hidden size is 256.

The loss function we use is negative log likelihood. The model reaches peak accuracy before 10 epochs.

### 6.3 Experiments

We conduct five experiments with the sequence to sequence model:

1. System response and API call prediction without context (Task 1 and 2)

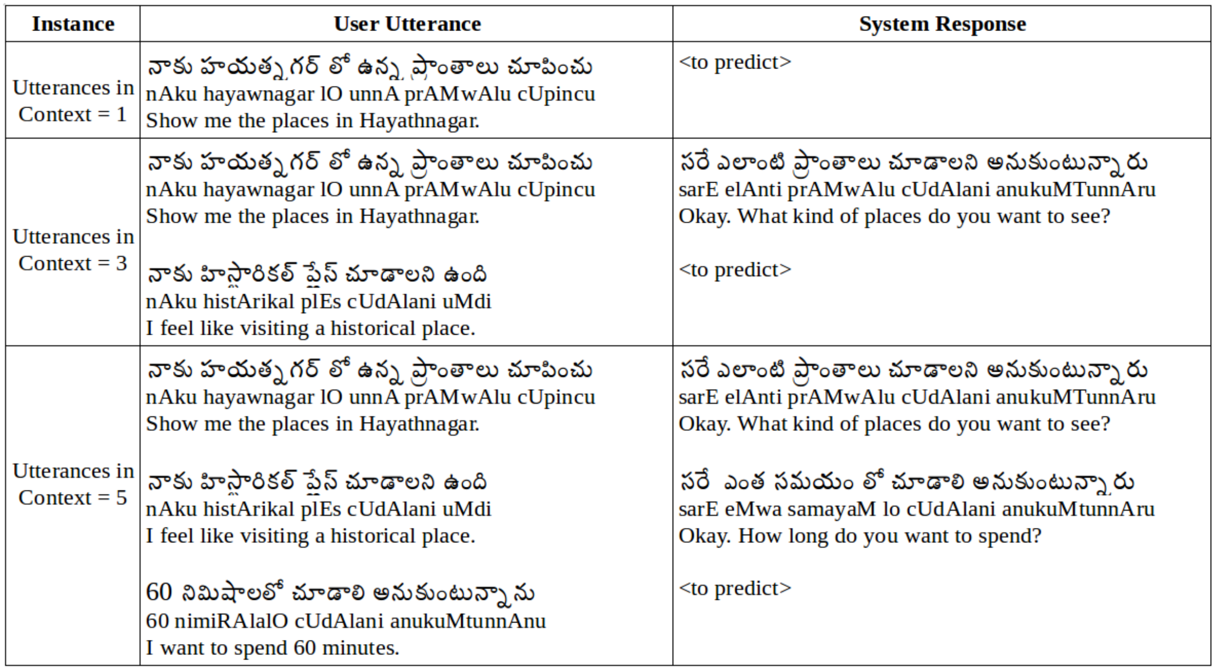
2. System response and API call prediction with context (Task 1 and 2)

3. System response prediction with context for, where only API call occurrence is predicted, not the API call itself (Task 1)

4. System response and API call prediction with context for updation (Task 1 and 3)

5. System response prediction with context for updation, where only API call occurrence is

[3]https://github.com/dandaprathyusha/End-to-end-dialog-system-for-telugu



| Instance | User Utterance | System Response |
|---|---|---|
| Utterances in Context = 1 | నాకు హయత్నగర్ లో ఉన్న ప్రాంతాలు చూపించు<br>nAku hayawnagar lO unnA prAMwAlu cUpincu<br>Show me the places in Hayathnagar. | <to predict> |
| Utterances in Context = 3 | నాకు హయత్నగర్ లో ఉన్న ప్రాంతాలు చూపించు<br>nAku hayawnagar lO unnA prAMwAlu cUpincu<br>Show me the places in Hayathnagar.<br><br>నాకు హిస్టారికల్ ప్లేస్ చూడాలని ఉంది<br>nAku histArikal plEs cUdAlani uMdi<br>I feel like visiting a historical place. | సరే ఎలాంటి ప్రాంతాలు చూడాలని అనుకుంటున్నారు<br>sarE elAnti prAMwAlu cUdAlani anukuMTunnAru<br>Okay. What kind of places do you want to see?<br><br><to predict> |
| Utterances in Context = 5 | నాకు హయత్నగర్ లో ఉన్న ప్రాంతాలు చూపించు<br>nAku hayawnagar lO unnA prAMwAlu cUpincu<br>Show me the places in Hayathnagar.<br><br>నాకు హిస్టారికల్ ప్లేస్ చూడాలని ఉంది<br>nAku histArikal plEs cUdAlani uMdi<br>I feel like visiting a historical place.<br><br>60 నిమిషాలలో చూడాలి అనుకుంటున్నాను<br>60 nimiRAlalO cUdAlani anukuMtunnAnu<br>I want to spend 60 minutes. | సరే ఎలాంటి ప్రాంతాలు చూడాలని అనుకుంటున్నారు<br>sarE elAnti prAMwAlu cUdAlani anukuMTunnAru<br>Okay. What kind of places do you want to see?<br><br>సరే  ఎంత సమయం లో చూడాలి అనుకుంటున్నారు<br>sarE eMwa samayaM lo cUdAlani anukuMtunnAru<br>Okay. How long do you want to spend?<br><br><to predict> |

Figure 5: Examples of Varying Context Length

predicted, not the API call itself (Task 1 and 3)

In the third and fifth experiments, we replace the full API call (api_call kanchanbagh historical 15) with just *api_call*, which the sequence to sequence model learns as a placeholder.

### 6.4 Results

| No. | Experiments | Acc. |
|---|---|---|
| 1 | Without Context + API calls | 59.8% |
| 2 | Context + API calls | 85.54% |
| 3 | Context Without API calls | 100% |
| 4 | Context + API calls | 79.67% |
| 5 | Context Without API calls | 100% |

Table 1: Sequence to Sequence Experiment Results

We can see from the results in Table 1 (the first column refers to experiment number) that the addition of context improves the accuracy. On analyzing the system utterances predicted during the second experiment, we saw that sequence to sequence learning is quite poor at predicting the required API calls. This is possibly due to the varying lengths in place names, etc. due to which the model is unable to predict all the components of an API call.

The third and fifth experiments which showed that the 14.46% error in the second experiment and the 20.33% error in the fourth experiment is mainly due to errors in predicting API calls, since their removal results in complete accuracy.

## 7 Predicting API calls using Memory Network

Since the sequence to sequence model performs poorly in API call prediction, we use memory networks to learn the same.

API calls are specific operations for information retrieval; we can consider predicting them as a simple classification problem. Memory networks are therefore more suitable for this task than a sequence-sequence model. They are unaffected by the varying lengths of place names unlike the sequence to sequence model.

The API call predicted by the Memory network is used to retrieve the required information from the knowledge base; typically a tourist location which fits the criteria in the query, with the highest rating.

### 7.1 Training and Testing Data

For Task 2 in memory networks, the input consists of a complete dialog history that has all the three fields of area, type and tour duration that are required for the completion of the query, in any order, up to, but not including, the *api_call* placeholder. The API call placeholder will be replaced by the API call predicted using the memory network.

In order to maintain an equal distribution of different combinations of queries (user specifies area first, user specifies type first, user specifies both area and type, user specifies none), we sample 24,000 instances of dialogs (6,000 of each type), of which we separate 20,000 instances (5,000 of each type) for training and 4,000 (1,000 of each type) instances for testing.

For Task 3, the input starts from the first predicted API call up to, but not including, the final *api_call* placeholder. The final API call placeholder will be replaced by the API call predicted using the memory network. For this task, we sample 15,000 instances for training and 3,000 for testing.

Task 4 is run only on memory network. The input starts from the last predicted API call up to the user's query for additional information. We train on 15,000 instances and test on 3,000.

## 7.2 Architecture

We use the architecture described by Sukhbaatar et al. (2015), which is primarily a recurrent neural network (RNN) which reads from an external memory before outputting a symbol. (See Figure 1)

A sequence of user ($U_i$) and system($S_i$) utterances $U_1, S_1, U_2 ..., S_{n-1}$ are taken as memory input and the last user utterance $U_n$, which is the query $q$, whose corresponding system response is an API call, is actual label $a$. The answer that will be predicted $a'$ by our model is the system response: API call.

In our model we are using layer-wise weight-tying where the input and the output embeddings are the same across different layers, i.e. $A^1 = A^2 = ... = A^K = A$ and $C^1 = C^2 = ...C^K = C$. The matrices $A, C$, of size $d \times V$, and the final weight matrix $W$, of size $V \times d$, are jointly learnt by minimizing a standard cross-entropy loss, where embedding dimension $d$ is 150.

The memory network reaches peak accuracy within 100 epochs for both tasks.

## 7.3 Results

In Table 2, Experiment 1 corresponds to Task 2, Experiment 2 corresponds to Task 3 and Experiment 3 corresponds to Task 4. Our accuracy for Task 2 and 3 in predicting API calls is on par with Bordes and Weston (2017). We perform better than Bordes and Weston (2017) in Task 4 since we do not add knowledge base facts, corresponding to the tourist location in the last API call, to the dialog history.

| No. | Experiment | Acc. |
|---|---|---|
| 1 | API calls | 100% |
| 2 | Updated API calls | 99.93% |
| 3 | API calls for Add. Info. | 100% |

Table 2: Memory Network Experiment Results

Since the data conforms to certain templates, the memory network performs very well. The accuracy is likely to drop for real world data.

## 8 Conclusion

We create a fairly large corpus of Telugu dialog data that can be used to train data-intensive models like neural networks, and can be used for further research.

We use the data to train a sequence-to-sequence dialog model that performs very well on predicting system responses, although it fares poorly with predicting API calls for information retrieval. Using memory networks we solve the API call prediction. We retrieve the highest rated tourist locations or other additional information from the knowledge base using the API call.

Our system is the only end-to-end dialog system to use deep learning methods in Telugu and proposes a better and more flexible model than the existing rule-based dialog system by Sravanthi et al. (2015; Reddy and Bandyopadhyay (2006), which can handle very few patterns. Our system is on par with the end-to-end dialog systems for English.

We have used a knowledge base in English and are able to predict API calls in English despite the dialogs being in Telugu. This means that we can use existing knowledge bases in English to build dialog systems in other low resource languages using similar methods of dialog data generation and deep learning.

## 9 Future Work

The system can be improved by introducing more initiative, for example, providing suggestions, taking negative responses into account, etc. The system should also be able to handle cases where no results are found for the user query by giving alternatives.

The system must also be trained with more varied data which has a greater number of patterns occurring; ideally on a sizeable corpus of natural dialog data created by native speakers. The system can then be tested subjectively through human evaluators.

## References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Antoine Bordes and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. *Proceedings of ICLR*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. 2016. Evaluating prerequisite qualities for learning end-to-end dialog systems. *Proceedings of ICLR*.

Rami Reddy Nandi Reddy and Sivaji Bandyopadhyay. 2006. Dialogue based question answering system in telugu. In *Proceedings of the Workshop on Multilingual Question Answering*, pages 53–60. Association for Computational Linguistics.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on empirical methods in natural language processing*, pages 583–593. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2015. Hierarchical neural network generative models for movie dialogues. *CoRR, abs/1507.04808*.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2016. Generative deep neural networks for dialogue: A short review. *arXiv preprint arXiv:1611.06216*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *Association for Computational Linguistics*.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 553–562.

Mullapudi Ch Sravanthi, Kuncham Prathyusha, and Radhika Mamidi. 2015. A dialogue system for telugu, a resource-poor language. In *CICLing (2)*, pages 364–374.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

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 Weston, Sumit Chopra, and Antoine Bordes. 2015a. Memory networks. *Proceedings of ICLR*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015b. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables. *arXiv preprint arXiv:1512.00965*.