Milestone 2 - Faithful Benchmark for Information Seeking Dialogue (Fact Hallucinations Detection and Prevention)

Jay Ashok Lal
jayashok@buffalo.edu

Shrishti Lakshmesh Karkera
karkera@buffalo.edu

Andrew Mikalsen
ajmikals@buffalo.edu

1 Introduction and Problem Description

Information Seeking Dialogue Agents serve the purpose of educating the User (information seeker) in form of natural conversation. For every prompt/user query, they have access to some knowledge text that is to be used to generate a response satisfying users information needs. In practice though, the responses generated by these agents contain information that is either absent in the knowledge or cannot be directly inferred from it. This problem is termed as 'Hallucination'.

In this work, we attempt to address the problem of Hallucination by learning to detecting it in existing responses and to create dialogue-agents that generate hallucination-free responses. The problem is largely broken into three subtasks, as described in the following sub-sections.

1.1 Task 1: Hallucination Critic

The hallucinated critic model should be able to determine whether the given response is hallucinated or not given the knowledge and history.

1.2 Task 2: BEGIN and VRM Multi-Class Multi-label Classification

The objective of task 2 is to distinguish between the different speech acts such as disclosure, question, acknowledgement, etc. given the knowledge and history and also identify between the response attribution classes such as hallucination, entailment, etc.

1.3 Task 3: Dialogue Generation

The dialogue generator should be able to respond with a response which is hallucination free (faithful to the knowledge and history), given the history and knowledge as inputs. In our baseline, we provide knowledge and prompt (most recent history) as inputs to the dialogue generator agent and it outputs a response which is faithful to the inputs.

2 Baseline Architecture

Our solution to the hallucination problem, involves training models for Hallucination detection (Task 1,2) and Prevention (Task3). Architectures for these tasks are described in the sub-sections that follow.

2.1 Tasks 1 and 2

To utilize both left and right context information, we opted to use Bi-LSTMs as the foundation of our baseline implementation for tasks 1 and 2. The baseline architecture is presented in Figure 1. We use a separate LSTM (i.e., different weights) for the response, knowledge, and history. To represent the history, we concatenate all entries together separated by a special separator token. We then use separate word embeddings to encode each of the response, knowledge, and history texts. Next, the hidden states of each of the three Bi-LSTMS (including both the forward and backward states) are concatenated together and passed as input to a feed-forward neural net. The output of the linear layer (which differs between tasks 1 and two; see the next paragraph) is then passed through a sigmoid function to generate the final output.

Since task 1 is a binary classification task, the output from the linear layer, and thus also the final output of the network, is a single scalar value. For task 2, the output is more complex since the classification is both multi-class and multi-label. In task 2, any combination of up to four labels from the BEGIN taxonomy (Hallucination, Entailment, Uncooperative, Generic) and up to five labels from the VRM taxonomy (Disclosure, Edification, Ack, Question, Advisement) may be assigned for a single input (i.e., a triplet consisting of the response, knowledge, and history). Thus, the output of the linear layer and final output of the network for task 2 is a nine-dimensional vector.

For task 1, we used original responses labeled as hallucinations as positive examples, and the new responses along with non-hallucination original

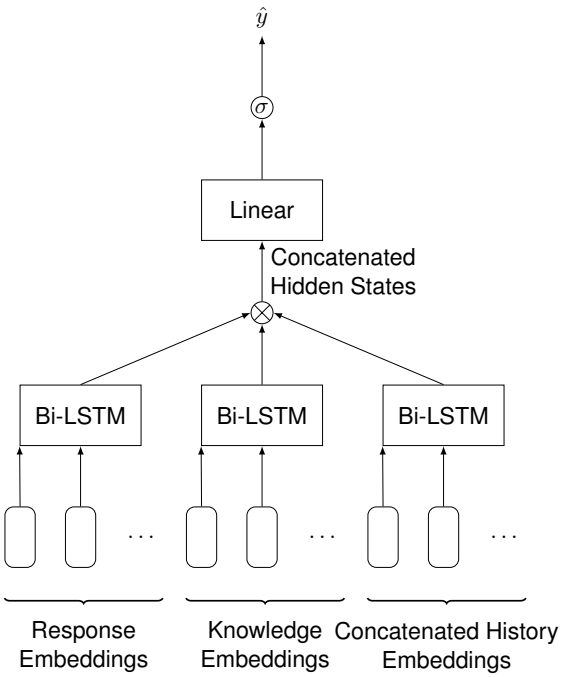

Figure 1: High-level architecture for tasks 1 and 2.

responses as negative examples. However, for task 2, BEGIN and VRM taxonomy labels are provided only for the original response, so training data was constructed only from positive examples.

2.2 Task 3: Dialogue Generation

Our baseline for the dialogue generator task is based on an encoder-decoder Sequence-to-Sequence (Seq2Seq) architecture. The encoder consists of two separate Bi-LSTM units, one for encoding knowledge text and the other for the prompt. Fig 2 shows the high-level architecture. The encoded hidden states from both these encoders are concatenated and a final context vector is generated that combines information from the knowledge and prompt. This context vector is used to initialize the decoder LSTM's hidden state. During inference, the Decoder generates text auto-regressively, one word at a time, taking in the previous output as current timestep input. During the training phase we use 'teacher-forcing' wherein half the time, current prediction is passed as next input, whereas remaining time the ground-truth prediction is passed on to next timestep as input. We use multi-class token-level CrossEntropy Loss and Adam Optimizer for training.

Table 4 shows the choice of hyperparameters estimated been emperically.

Table 1: Task 1 accuracy and macro-averaged F1 score on the test split for models trained using just the response (R); the response and the knowledge (R+K); and the response, knowledge, and history (R+K+H).

	Accuracy (%)	F1-Score
R	82.90	0.79
R+K	84.31	0.80
R+K+H	85.22	0.81

Table 2: Task 2 accuracy and macro-averaged F1 score on the test split.

Accuracy (%)	F1-Score
87.77	0.78

3 Baseline Results

The results for each of the baseline models are described in the following sub-sections.

3.1 Task 1: Hallucination Critic

Table 1 shows the accuracy and macro-averaged F1 score of the task 1 classifier when using only the response; using the response and the knowledge; and using all of the response, knowledge, and history. We use the standard definitions of accuracy and F1-score for binary classification.

3.2 Task 2: BEGIN and VRM Multi-Class Multi-label Classification

For task 2, results are given in Table 2 (accuracy and macro-averaged F1 score) and in Table!3 (confusion matrix). We define a true positive on the granularity of a label, counting the assignment as a true positive when the input was correctly assigned the label as one of its labels. We use the corresponding definitions for false positive, true negative, and false negative. This is one of multiple standard ways of defining these metrics for multi-label classification.

3.3 Task 3: Dialogue Generation

Table 5 and 6 depict the quantitative results of the baseline architecture.

Table 3: Task 2 confusion matrix on the test split.

	Negative	Positive
True	19,752	1,036
False	2,186	3,342

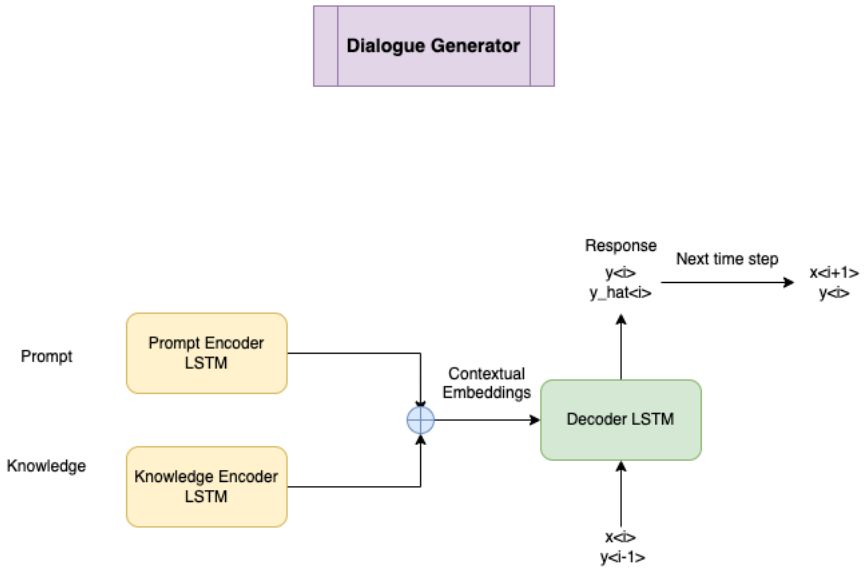

Figure 2: High-level architecture for task-3 response generation.
Prompt is the last text from history, i.e user query

Table 4: Task 3 - Seq2Seq Architecture Hyperparameters.

	Parameter
Vocabulary Size	12000
Embedding-Dimension	300
Knowledge Encoder-hidden units	512
Knowledge Encoder-Layers	2
Prompt Encoder-hidden units	512
Prompt Encoder-Layers	2
Response Decoder-hidden units	512
Response Decoder-Layers	4
Dropout	0.5
Weight-Decay	1e-4

Table 5: Task 3 Evaluating quality of generated response.

	Mean Score
Bleu	12e-9
Rouge-1	0.198
Rouge-2	0.005
Rouge-L	0.177
Bert	0.062

Table 6: Task 3 Measuring Hallucination in generated response. (Using Task-1 Model)

	Values
Mean Predicted Score	0.202 (Not Hallucination)
Standard Deviation	0.09

4 Analysis and Areas for Improvement

4.1 Task 1: Hallucination Critic

We see that the network scores decent accuracy for just the response. This suggests that specific wordings of the responses are correlated with hallucinations and non-hallucinations. Intuitively, this makes perfect sense. Responses that are hallucinations will likely contain phrases where the wizard is talking about themselves and their own hallucinated experiences, i.e., “I remember ...//", “My favorite ..", "I enjoy..." etc, which will probably be absent from knowledge-grounded responses. This is however a very blunt approach to hallucination detection, and it’s expected that incorporating the history will vastly improve performance.

Unfortunately this does not appear to be the case for our current model. We see only very small improvements in accuracy and F1-score when incorporating the knowledge. This suggests that as-is, the network is not learning the relationships between the knowledge and the response (or lack thereof) for hallucinations vs. non-hallucinations. Thus, the primary directions moving forward should be focused on improving this aspect. It isn’t clear yet how important history is, but this is another area that might require improvements. Its possible that history becomes important in some instances that are grounded in both the history and the knowledge, that might otherwise look like hallucinations if the history is ignored.

The current architecture creates a hidden representation for the response and the knowledge, then uses this representation in a linear network to see how grounded the response is in the knowledge. Right now, there is not any pooling done for the intermediate outputs of the LSTM. This is a technique that will be interesting to try. Another observation is that the knowledge has a lot of "fluff" that might not be replicated in the actual response. Perhaps learning how to filter out this fluff during pooling could improve performance. We also observe that transformers are the state-of-the-art even for classification tasks. Another promising approach, which was recommended in the project write up, is to use multi-encoder pipelines to encode this information. We also want to try other models, i.e., pretrained transformers, and see if we can get improved performance out of them.

4.2 Task 2: BEGIN and VRM Multi-Class Multi-label Classification

We believe that the observations and avenues for improvement discussed above for task 1 also apply to task 2. Additionally, our data showcases some challenges specific to task 2 we wish to address. First, we note that the test split has 30,000 entries. In the confusion matrix, we see that the vast majority of classifications are true negatives. This is likely because labels are sparsely assigned, i.e., most inputs are only assigned a few of the possible labels. We see that there are many more false positives than true positives. The implication is that the model learns to infrequently assign labels. However, when the model decides to try to assign a label, it's usually the wrong one. As suggested in the project writeup, we want to try using linguistic features to encode the relevant information to improve the true positive rate.

4.3 Task 3: Dialogue Generation

We observe from Table 5, 6 that, even though all the responses are classified as hallucination-free, the generated response quality is very poor. Upon closer inspection we notice that all the predicted responses are similar to *"I do'nt know that the, the the, the."*. Further during the training phase, we also noticed that the model was overfitting - validation loss being higher than the training loss. However, no amount of regularization (See table 4) or parameter reduction was able to yield better results.

One interpretation of this would be that the task of response generation is difficult to learn from FaithDial dataset alone. This might be because generation implicitly requires rephrasing the knowledge text, and thus understanding of the language semantics. Hence, it might be better to use a pretrained language model such as GPT-2 as a decoder.

Morever, the lack of variation in responses, also indicates that the two BiLSTMs are unable to properly encode knowledge and prompt. Perhaps a pretrained encoder like BERT might be better able to capture the context from input text pieces, or attention can be deployed to obtain a dynamic encoding of the context.

We intend to incorporate these changes and conduct more experiments for Milestone 3.

5 Contributions

Jay worked on task 3. Shrishti worked on task 3. Andrew worked on tasks 1 and 2.