Answering Student Queries with a Supervised Memory Conversational Agent

Florian Baud, Alex Aussem

Answering Student Queries with a Supervised Memory Conversational Agent

Florian Baud, Alex Aussem
Université Lyon 1, LIRIS UMR 5205 CNRS, France
florian.baud@liris.cnrs.fr, alexandre.aussem@liris.cnrs.fr

Abstract

This paper describes a discussion-bot that provides answers to students' questions about the Data Science master program at the University of Lyon 1. Based on a seq2seq architecture combined with a supervised memory module, the bot identifies the questioner's interest and encodes relevant information from the past conversation to provide personalized answers. A dialogue generator based on hand-crafted dialogues was built to train our model on these synthetic dialogues. The agent and its memory are adaptable to another context by modifying the intention database of the generator. The model was deployed and the results show that the discussion-bot meets most students' learning requests. We discuss further directions that might be taken to increase the model's effectivenes.

Introduction

Every year, students in their final year of a bachelor's degree are looking for a master's degree that best suits their profile and expectations. Most of them have many questions (e.g., requirements, prerequisites, applications, deadlines, and campus life) about their future training. At the university of Lyon 1, a conversational agent was developed to answer any request for information, in a personalized way, from candidates for the Data Science master's degree. As many of these questions tend to be asked every year, the bulk of these pre-admission queries would be addressed through the agent itself. This paper presents the technical aspects and the implementation of this conversational agent. The agent can give personalized responses through memory that it updates as the conversation goes along.

As students' queries may depend on their past utterances that the bot should keep in mind, we focus our attention on so-called dialog state tracking systems (Shukla et al. 2020), which map values for specific slots (for example student-university). The proposed agent takes into account the past of the conversation with a memory module that detects features for selecting the correct response. The memory used in our work is a "simple" binary vector that encodes the student "profile", that is, all the attributes required to correctly answer their questions (Fig 1).

To train our model, we collected data from the previous school year. We asked students to write questions about the data science master program, we then answered all these questions. After cleaning and checking these questions, we built a dialog generator that mimics proper discussions between prospective students and the conversational agent. The dialog flow logic is quite simple for questions that don't require any information about the student profile. For questions requiring specific information about the prior cursus of the student, the bot has to request this information in natural language from the student, which complicates the dialog

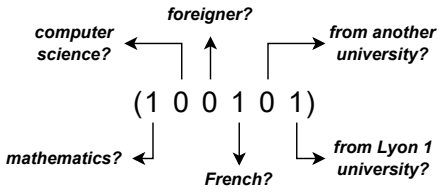

Figure 1: Six attributes characterize a student in order to properly answer his questions. In this example, the student is French, from Lyon 1 University, wishing to be enrolled on the mathematical program of the Data Science master.

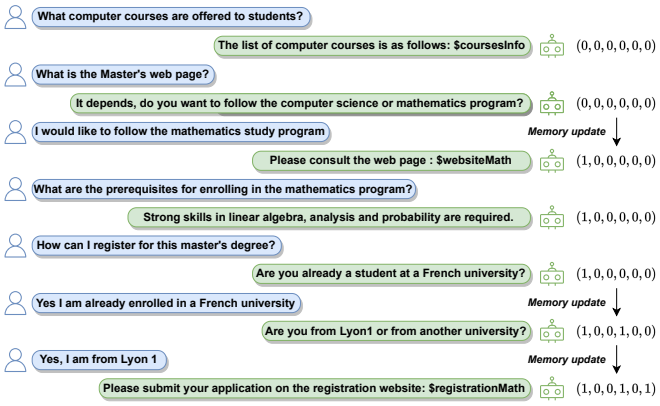

Figure 2: A typical dialog between a student (colored in blue), and the bot (in green). The memory vector, shown on the right hand-side, is updated in the course of the chat as the bot learns the specific profile of the student.

generation process. In this paper, we propose a dialog generator that takes into account the dialogue states.

The conversational model consists of three modules: a natural language understanding module, a dialog manager, and a natural language generator. These modules are often part of a conversational agent (Shukla et al. 2020). Our conversational agent aims to be simple, interpretable and easily deployed. We use a neural network architecture where a Gated Recurrent Unit (GRU) (Cho et al. 2014) acts as a natural language understanding module. Our dialog manager is a single-head cross-attention between the past conversation memory of student details and the currently encoded utterance. The response generation is a simple feed-forward network that outputs the response class and updates the memory.

We first discuss the acquisition and the generation of training dialogs. We then describe a neural conversational agent model able to answer students wishing to enroll in a master program. Finally, we report on the experiments performed to evaluate the practical usefulness of the model to guide and inform the prospective Master students. Code and data are available on github [1].

Related Work

We start with a brief review of related work on conversational agents applied to the education domain. In (Feng et al. 2006), the authors discuss the use of a chatbot to answer students' questions on a forum where some questions are already answered. Their goal is to respond to students using this forum without resorting to a search engine, which they believe would discourage interchanges. They retrieve a set of semantically-related passages that match a student's interest by directly computing the cosine similarity between question post and archived data using the *TF-IDF* technique (Salton 1989) in order to find the answer in the forum thread.

(Aujogue and Aussem 2019) proposed a model for multi-turn response generation that is end-to-end trainable. Its hierarchical structure, allows the model to focus on both words and sentences of the past conversation in order to correctly respond to a new utterance. However this model is end-to-end, the memory is not clearly defined and supervised. Moreover, the answers are generated word by word at the risk of producing badly formulated answers.

A more recent approach (Santana et al. 2021) that uses *Transformers* (Vaswani et al. 2017) to detect both user intent and named entities to answer students' administrative questions. This approach shares some similar intuition with our proposed approach, however our bot is endowed with a supervised binary memory vector to capture the useful information of the student profile - updated in the course of the conversation - in order to answer questions in a personal way.

There is also an ongoing discussion about which framework to use for building a chatbot for the educational domain. (Rooein, Paolini, and Pernici 2022) pointed out that making chatbots customizable enhance students' learning experience. Indeed, answering the students in a personal way will improve their general experience. Our task is however somewhat different : answering various queries of prospective students about a Master program. (Sonderegger and Seufert 2022) propose a framework and address several limitations regarding data privacy and dependence on big technology suppliers. As the informative data may change over the years, our chatbot should be customizable enough to add new data.

Our contributions are two-fold: first, we propose a neural conversational agent with a simple memory design that is controllable and intelligible. Second, we developed a dialog generator with memory that can be expanded and updated depending on the context.

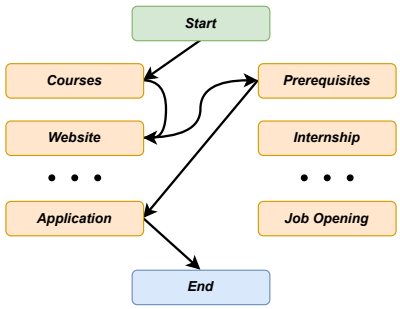

Figure 3: The generated dialogs consist of a random walk through various topics.

Dialogs generator

There are usually very few public data available related to the specific task at hand that can be used to create a conversational agent with a specific goal. Thus the datasets are often created by hand to train a mutliclass classifier where each intent is a class. Our case is slightly different because bot answers are not only based on the immediate past input from the student. We need to take into account the student's information to correctly answer his query. This information is to be captured in the course of the conversation, otherwise the chatbot has to ask the student for it when needed.

Our data generator (Algorithm 1) is designed to simulate such conversations and generate a memory vector associated with these exchanges. The memory vector encodes the student's properties in the form of a binary vector, each dimension of the vector representing a binary attribute (Fig. 1). This vector is updated in the course of the conversation, therefore it is fed as input to our model and potentially updated according the information provided by the current user utterance. This way of representing the memory is advantageous because it is fully supervised and we may easily add new attributes if necessary.

However, the developer needs to define explicitly all the attributes. In contrast, end-to-end models like the one discussed in (Aujogue and Aussem 2019) use a more complex "Hierarchical Attention Network" learns to capture the relevant information from the conversation context, without any help. The conversation context is seen as a hierarchical struc-

[1] https://github.com/florianbaud/chatbot-lyon1

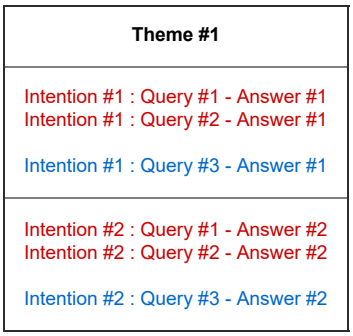

Figure 4: Example of topic format where red intentions are for train dataset and blue one for test dataset. An intention database has multiple topics like this.

Algorithm 1 Dialogs Generator

Require: $N \geq 0$ ▷ Length of the dialog
Require: $Topics$
Require: $Students' profile$
  $D \leftarrow [\,]$ ▷ List of utterances
  $M \leftarrow [\,]$ ▷ List of memory state
  while $N \neq 0$ **do**
    $Topic \leftarrow$ Random($Topics$)
    $User$, $Bot \leftarrow$ Random($TopicUtterances$)
    D.append($User$) ▷ Append $User$ utterance
    if memory required **then**
      for $BotQuestion$ to ask **do**
        if $UserAnswer$ not in memory **then**
          D.append($BotQuestion$)
          D.append($UserAnswer$)
          Update(M) ▷ Update memory
        end if
      end for
    end if
    D.append(Bot) ▷ Append Bot utterance
    $N \leftarrow N - 1$
  end while

ture (words form an utterance, and utterances form the context) and has two levels of sequential relationships among both words and utterances within the structure. Such model (Aujogue and Aussem 2019) is very appealing but it is far more complex and require much more data to train.

The generator (Algorithm 1) takes as input the desired dialog length, the input topics, and a student's profile. We built topics by grouping data collected from previous school years, where we asked students to write down ten questions-responses. Each pair of question-response was assigned to a theme within which pairs are normalized to have the same answer (see Fig.4). These topics, which form an intention database, have the advantage of being editable so that a non-technical agent can add a new couple of question-response. The students' profile is an input of the generator and is part of the model memory. Finally, the memory or profile consists of six binary attributes that are depicted in Fig.1.

Before generating a new dialog, we draw a profile at random. The algorithm starts to initialize the memory and dialog list. The initial memory is a null vector that the algorithm will update during the generation. Then, during the dialog generation, the algorithm explores a list of topic (Fig.3) and draw a pair at random. The algorithm has to update the memory depending on the pair drawn. When memory is necessary for answering the question, the algorithm has to check the memory and ask questions about corresponding missing memory information. Finally, the resulting dialog consists of pair and memory lists stored in text format and then used as training dialog.

In order to test the model, the generator can use two disjoint ensembles of intention when drawing at random an intention. This way, the model has an unseen set of intentions during the testing procedure. In our experiments, we generated 20.000 train dialogues of a maximum length of 35 and 2.000 test dialogues of a maximum length of 35.

Model description

Our conversational agent model is a neural network composed of a GRU (Cho et al. 2014) that takes care of encoding the user's sequence, a single head cross attention layer incorporating the binary memory vector and a feed forward network to obtain the new memory state and the user's intention. This model is rather simple, has few parameters, and the memory is simple and meaningful.

Assume a conversation of length M between the user and the conversational agent, the user's sentence $u_j,\ j \in [0, M]$ is the jth sentence of the user and each sentence u_j has a sequence of tokens $x_{ij},\ i \in [1, N_j]$ defined by a vocabulary ν. When a token is not present in the vocabulary, it is replaced by the unknown token. The learning task is to determine the correct intention c_j and the next state of the memory m_j.

Firstly the set of tokens of the jth sentence are converted into a sequence of vectors using an embeddings matrix E, so each x_{ij} is associated with $e_{ij} = E(x_{ij})$. Embeddings are trained at the same time as the model is. We compute the representations of each token in the sentence u_j with a GRU:

$$\overrightarrow{h}_{ij} = \overrightarrow{GRU}(e_{ij}, \overrightarrow{h}_{i-1,j}),\ i \in [1, N_j]$$
$$\overleftarrow{h}_{ij} = \overleftarrow{GRU}(e_{ij}, \overleftarrow{h}_{i+1,j}),\ i \in [N_j, 1]$$

The GRU module is bidirectional therefore each token takes into account all the rest of the sentence with $H_{ij} = [\overrightarrow{h}_{ij}, \overleftarrow{h}_{ij}],\ i \in [1, N_j]$, the global meaning of the sentence is given by $H_{u_j} = [\overrightarrow{h}_{N_j j}, \overleftarrow{h}_{1j}]$, where the operation $[\cdot, \cdot]$ corresponds to the concatenation.

Then the representation H_{u_j} is concatenated with the memory vector m_j of the state j, $s_j = [H_{u_j}, m_j]$. A dot product is computed between s_j and W_s then, b_s is added. The result is fed to a $\tanh$ function, i.e. $S_j = \tanh(s_j \cdot W_s + b_s)$, where W_s and b_s are trainable parameters. Next we compute a single head cross attention to obtain a vector C_j weighting the tokens H_{ij} with the context S_j as described in (Bahdanau, Cho, and Bengio 2015) and (Luong, Pham, and Manning 2015) :

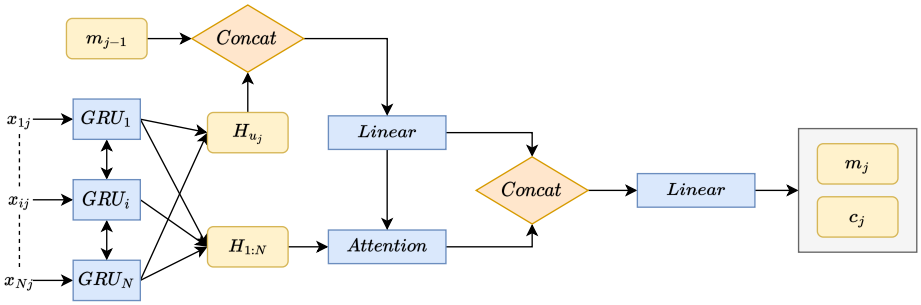

Figure 5: The user's statement at time j is encoded by a bi-GRU, then we derive two states H_{u_j} and $H_{1:N}$. The first one represents the general meaning of the user's utterance, the second one is the set of hidden states of the bi-GRU corresponding to the meanings of each token. They are processed by an attention mechanism where the values are the $H_{1:N}$ and the query is the concatenation of H_{u_j} and the memory m_j. The latter and the result of the attention mechanism determine the new memory state and the user's intention.

$$\tilde{\alpha_{ij}} = \text{score}(S_j, H_{ij})$$
$$\alpha_{ij} = \frac{\exp(\tilde{\alpha_{ij}})}{\sum_{i'} \exp(\tilde{\alpha_{i'j}})}$$
$$C_j = \sum_{i=1}^{N_j} \alpha_{ij} \cdot H_{ij}$$

The score function is a feedforward neural network as used in (Bahdanau, Cho, and Bengio 2015) : $\text{score}(x, y) = x \cdot W_{\text{score}} \cdot y$, where W_{score} is a trainable parameter.

Finally the user's intention and memory are obtained with another fully connected layer by concatenating C_j and S_j and then computing $R_j = \sigma([C_j, S_j] \cdot W_r + b_r)$ where σ is the sigmoid function, W_r and b_r are trainable parameters, and $R_j \in \{0, 1\}^{|m_j|+|c_j|}$. We separate memory and intent from R_j, giving $R_{m_j} = R_{j,1:|m_j|}$ for memory and $R_{c_j} = R_{j,|m_j|:|m_j|+|c_j|}$ for user intent. For training our model, we define the following two losses :

$$L(x, y) = -\sum(y \cdot \log(x) + (1-y) \cdot \log(1-x))$$
$$L_{m_j} = L(Y_{m_j}, R_{m_j})$$
$$L_{c_j} = L(Y_{c_j}, R_{c_j})$$

Where Y_{m_j} is the target vector of the memory state and Y_{c_j} is the user's intention target vector. Our model is trained with a mixture of these losses $L_j = \beta L_{m_j} + L_{c_j}$, where β is a hyperparameter controlling the memory importance. Parameters are updated by backpropagation of the gradient with the Adam (Kingma and Ba 2014) optimization function. Knowing c_j and m_j, we return the textual answer to the user by looking in a matching table.

The size of the embedding vectors is 128 and the hidden layer of the GRU module, the W_r matrix of R_j has a size of 384 because of the bi-GRU. We use a dropout (Srivastava et al. 2014) rate of 0.2 for the GRU and 0.1 for the cross-attention. Our final model has only one GRU layer and has a total of 586,610 trainable parameters, In comparison BERT (Devlin et al. 2019) has 110M and 340M parameters for $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$ respectively only for the encoder.

Experiments

We trained our model on GPUs freely provided by *Google Colab*. Our model is assessed in two different ways: the first one consists in using the synthetic dataset provided by the dialog generator, the second one is based on a human evaluation where we invited students wishing to apply to our master programm to use the conversational agent.

Experiment	Correct	Wrong	Misunderstood
Test set	48,23	19,30	32,47
Deployment	36,11	6,22	57,67

Table 1: The results are percentages of the number of discussions processed in the experiments.

The test set contains pairs that the conversational agent has never seen in the training set. The wording of the sentences is therefore very different, some words do not appear in the training set. The model achieved a performance on this dataset of 48.23% of correct answers, 19.30% of wrong answers and 32.47% of answers where the bot was not able to understand the question and asked the user to rephrase it. These results are imperfect because only half of the answers are correct, but when the agent cannot give the right answer, it is preferable to ask the user for a reformulation.

In the second experiment, we trained our model with both training and test dataset. Then the chatbot was deployed[2] and used by the students in a real situation. After several months, we collected discussions from production. We then checked the answers of the conversational agent by hand. About 200 discussions were analyzed and in total 1479 answers were given by our model, among these 6.22% were wrong answers, 36.11% were correct answers and in 57.67% of the cases the agent asked for a rephrasing. Concretely, users tend to rephrase their question, so the conversational agent has an additional chance to answer correctly.

[2]http://chatbotinfo.univ-lyon1.fr/

Conclusions and future work

We discussed a Seq-to-Seq based conversational agent for students applying for the Master's program in Data Science that was deployed at the University of Lyon 1. The conversational agent answers in a natural way and is able to request further information from the user to give a personalized response with our memory module. Our model with memory is adjustable to any context by using another generator's database. In addition, we proposed a dialog generator that handles dialog states and generates proper conversations for training our conversational model. Nevertheless, one third of the questions are not always well understood by the conversational agent. There are several interesting directions for future work, for instance the integration of more sophisticated linguistic models like BERT that seem ready in production settings.

References

Aujogue, J., and Aussem, A. 2019. Hierarchical recurrent attention networks for context-aware education chatbots. In *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, 1–8. IEEE.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In Bengio, Y., and LeCun, Y., eds., *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. Doha, Qatar: Association for Computational Linguistics.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. Minneapolis, Minnesota: Association for Computational Linguistics.

Feng, D.; Shaw, E.; Kim, J.; and Hovy, E. 2006. An intelligent discussion-bot for answering student queries in threaded discussions. In *Proceedings of the 11th International Conference on Intelligent User Interfaces*, IUI '06, 171–177. New York, NY, USA: Association for Computing Machinery.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412–1421. Lisbon, Portugal: Association for Computational Linguistics.

Rooein, D.; Paolini, P.; and Pernici, B. 2022. Educational chatbots: A sustainable approach for customizable conversations for education. In Cukurova, M.; Rummel, N.; Gillet, D.; McLaren, B. M.; and Uhomoibhi, J., eds., *Proceedings of the 14th International Conference on Computer Supported Education, CSEDU 2022, Online Streaming, April 22-24, 2022, Volume 1*, 314–321. SCITEPRESS.

Salton, G. 1989. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. USA: Addison-Wesley Longman Publishing Co., Inc.

Santana, R.; Ferreira, S.; Rolim, V. B.; de Miranda, P. B. C.; Nascimento, A. C. A.; and Mello, R. F. 2021. A chatbot to support basic students questions. In *LALA*.

Shukla, S.; Liden, L.; Shayandeh, S.; Kamal, E.; Li, J.; Mazzola, M.; Park, T.; Peng, B.; and Gao, J. 2020. Conversation Learner - a machine teaching tool for building dialog managers for task-oriented dialog systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 343–349. Online: Association for Computational Linguistics.

Sonderegger, S., and Seufert, S. 2022. Chatbot-mediated learning: Conceptual framework for the design of chatbot use cases in education. In Cukurova, M.; Rummel, N.; Gillet, D.; McLaren, B. M.; and Uhomoibhi, J., eds., *Proceedings of the 14th International Conference on Computer Supported Education, CSEDU 2022, Online Streaming, April 22-24, 2022, Volume 1*, 207–215. SCITEPRESS.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15(1):1929–1958.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.