https://doi.org/10.52326/ic-ecco.2021/CS.06

Using Luong's attention mechanism and simple classifiers to make people overcome psychological illnesses

Vasile Păpăluță
Technical University of Moldova, Faculty of Computers, Informatics, and Microelectronics,
Department of Software Engineering and Automatics, Chişinău, Republic of Moldova
papaluta.vasile@isa.utm.md

***Abstract*—Conversational AI is the set of technologies behind automated messaging and speech-enabled applications that offer human-like interactions between computers and humans. It can communicate like a human by recognizing speech and text, understanding intent, deciphering different languages, and responding in a way that mimics human conversation. The objectives of this research are to explore the applicability of conversational AI technology in creating a chatbot for assisting people struggling with psychological illnesses and mental dysfunctions.**

The main hypothesis is that having an NLP system containing an NLG submodule (module for generation of the Natural text) and an NLU submodule (module for recognizing the emotional state of the person using this chatbot. We use an NLU submodule because we can't rely only on the artificially generated text as a response for a person in an awful emotional state. Even more, we can use the information from the NLU submodule for stronger strategies generation to ensure emotional support.

The system represents a chatbot with two NLP modules, Natural Language Generation, being represented by a Seq2Seq Neural Network with the Loung's attention mechanism, and a Natural Language Understanding module represented by a classical classification NLP Pipeline that classifies the text in multiple emotional state classes. To interact with the user it uses the Telegram API and is able to save the user messages and the chatbot answers into a simple SQLite Data Base.

Even if this implementation wouldn't replace the real psychologists, with accurate management and maybe with additional inputs for professionals in psychology it may become a tool for detecting people with possible psychological and mental illnesses which can become the first step in further therapy with a real psychologist.

Keywords—Neural Networks; Natural Language Processing; Python; PyTorch; Loung's attention mechanism

I. INTRODUCTION

In today's world full of high speed and stress, the psychological state of people has become more affected by psychological disorders. Even if going to a psychologist isn't a big deal, people are ashamed of this, especially men. In the era where people have a lot of trust in written software and Artificial Intelligence, a lot of technological solutions to this problem appeared. Woebot[1], Moodnotes[2], Wysa[3], Youper[4] just mention some, are chatbots oriented on helping and assisting people with psychological disorders.

This work is exploring the implementation in more technical details of a chatbot created to assist people with psychological disorders. The initial problem that this solution is addressing was that people who are struggling with bad feelings or psychological disorders don't always have enough courage to address them to a psychologist. That's why a chatbot that can handle a dialog with a person and at the same time can detect the emotional and psychological state of the person by how it is chatting and direct the communication in the direction that will bring the person to a more positive state was created.

II. SOLUTION

A. General description.

The system is constructed from the following elements: the telegram API (Initially, the first version of the chatbot was built on the telegram platform), the text classification pipeline, the emotional state to response mapper, the seq2seq model for text generation, and the database. The general scheme of the interaction between all modules of the chatbot is listed below:

https://doi.org/10.52326/ic-ecco.2021/CS.06

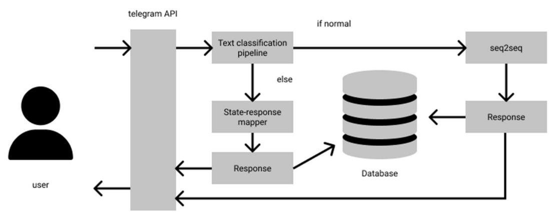

Figure 1: The architecture of the chatbot.

B. *Telegram API*

The telegram platform offers a big range of possibilities to create a chatbot. However, for this project, was used the simplest way to do this - get requests. All interaction with the telegram is stored in the telegram_bot class, which has the access token and the function for the interaction with it. Below are listed the links for the API interaction:

The endpoint for getting the messages sent to the bot:

https://api.telegram.org/bot<token>/getUpdates?timeout=100&offset=<offset>

The endpoint to send messages from the bot:

https://api.telegram.org/bot<token>/sendMessage?chat_id=<chat_id>&text=<msg>

These endpoints are used by the chatbot to interact with the telegram users, however, it can be adapted to other messaging platforms too.

C. *The data sources*

As was said the architecture of the chatbot includes 2 models: one for text classification and one for text generation. Each model had its data source. For the text generation model, was used the industry-standard - Cornel Movie-Dialog Corpus[5]. It contains a large metadata-rich collection of fictional conversations extracted from raw movie scripts, and it doesn't contain any sentiment label.

The emotional state classification model, were extracted through the Reddit API[6] reddits from the following subreddits: rape, PTSD, anxiety, abuse, depression, and bullying. By adding 800 additional ham samples from the SMS Spam Collection Data Set[7] the data set for the emotional state classification task was created. Below are listed the number of samples per class:

Table 1: The number of samples of every class of emotional state classifier.

Class	Number of samples
rape	878
ptsd	811
normal	800
anxiety	720
abuse	709
depression	622
bullying	610

D. *The text classification pipeline*

Unfortunately, we cannot pass simple text to a machine learning model. The text should be preprocessed and represented in an optimal numerical form. Usually, for that, an NLP pipeline is constructed. The pipeline is illustrated below:

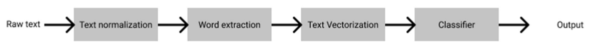

Figure 2: A classic NLP classification pipeline.

The first step in this pipeline is the text normalization step. It is represented by bringing the text to lowercase, eliminating the special symbols like newline and additional spaces. In our case, an additional step was added: erasing from text everything that isn't text.

The second step is the word extraction step. Following Zipf's law, we must extract the most frequent tokens (a.k.a. stopwords) which we can obtain from the nltk library[8], and the hapaxes - the tokens that we find only once in the whole corpus.

The next step is text vectorization. During the vectorization step, we try to represent text in a vectorial form. One of the most popular ones is the TF-IDF one. TF-IDF vectorization was chosen because this representation gives low scores to words that are very frequent throw the corpus, lowering in such a way their impact on the model, and giving higher scores to the more rare tokens.

The final module of the pipeline is the Machine Learning model itself - in our case, it is the Logistic Regression model. It gave the highest accuracy. The accuracy of other models are listed below:

Table 2: The accuracy of the different pipelines.

Model	Accuracy
Bernoulli Naive Bayes	0.511

https://doi.org/10.52326/ic-ecco.2021/CS.06

Multinomial Naive Bayes	0.440
Logistic Regression	0.740
Support Vector Machines	0.726
Decision Tree	0.581
Random Forest	0.677

The text classification pipeline is returning one of the classes listed in table nr. 1. If the returned class is 'normal' then the response is generated by the text generation model, else it is chosen from the emotional state - response mapper.

E. *The emotional state - response mapper*

To be sure that the user of the chatbot is getting the messages that he or she needs in her emotional state, for every non-normal emotional state were created a list of 10 predefined responses. To add more interactivity whenever a non-normal emotional state is triggered a random response from its list of responses is chosen as a response.

F. *The text generation model*

The text generation model is represented by a seq2seq model. The goal of the seq2seq model is to take a variable-length sequence as an input and return a variable-length sequence as an output using a fixed-sized model.

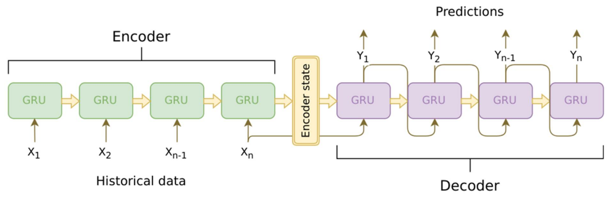

Figure 2: The seq2seq architecture.

Sutskever et al.[9] discovered that this can be achieved by using two separate recurrent neural networks together. One RNN acts as an encoder, which encodes a variable lengths input sequence to a fixed-length context vector. In theory, this context vector will contain semantic information about the query sentence that is input to the bot. The second RNN is a decoder, which takes an input word and the context vector, and returns a guess from the next word in the sequence and a hidden state to use in the next iteration.

At the heart of our encoder is a multi-layered Gated Recurrent Unit, invented by Cho et al.[10] in 2014. In this project a bidirectional variant of the GRU, meaning that there are essentially two independent RNNs: one that is fed the input sequence in normal sequential order, and one that is fed the input sequence in reverse order. The outputs of each network are summed at each time step. Using a bidirectional GRU will give us the advantage of encoding both past and future contexts.

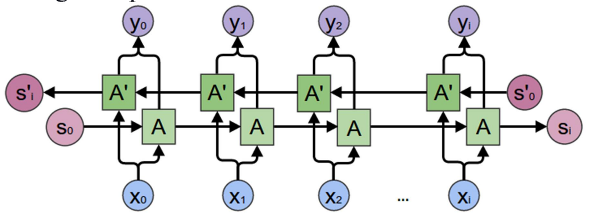

Figure 3: Bidirectional RNN.

The decoder RNN generates the response sentence in a token-by-token fashion. It uses the encoder's context vectors, and internal hidden states to generate the next word in the sequence. It continues generating words until it outputs an EOS (End Of Sequence) token. A common problem with a vanilla seq2seq decoder is that if we rely solely on the context vector to encode the entire input sequence's meaning, we will likely have information loss. This is especially the case when dealing with long input sequences, greatly limiting the capability of our decoder.

To combat this, Bahdanau et al.[11] created an “attention mechanism” that allows the decoder to pay attention to certain parts of the input sequence, rather than using the entire fixed context at every step.

Luong et al.[12] improved upon Bahdanau et al.’s groundwork by creating “Global attention”. The key difference is that with “Global attention”, we consider all of the encoder’s hidden states, as opposed to Bahdanau et al.’s “Local attention”, which only considers the encoder’s hidden state from the current time step. Another difference is that with “Global attention”, we calculate attention weights, or energies, using the hidden state of the decoder from the current time step only. Bahdanau et al.’s attention calculation requires knowledge of the decoder’s state from the previous time step. Also, Luong et al. provide various methods to calculate the attention energies between the encoder output and decoder output which are called “score functions”:

$$\operatorname{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_{\boldsymbol{a}} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_{\boldsymbol{a}}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

Figure 4: The score function of Loung’s model.

https://doi.org/10.52326/ic-ecco.2021/CS.06

Overall, the Global attention mechanism can be summarized by the following figure:

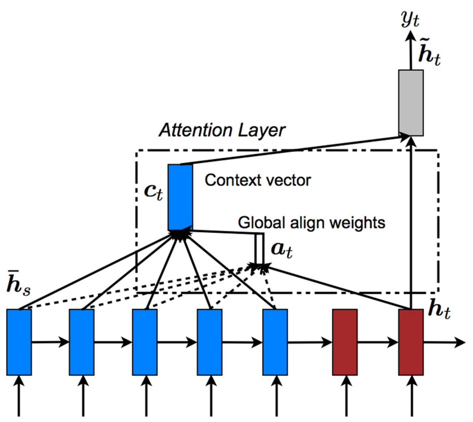

Figure 5: The Global attention mechanism.

The last step for an operating seq2seq model is the Greedy decoding. Greedy decoding is the decoding method that we use during training when we are not using teacher forcing.

G. The database

The last module of the Chatbot is the database. It has only 2 tables: messages and users. The first one is used to store the data about messages received by the chatbot, while the second one store the information about users that interact with it. When a new user is sending messages to the chatbot it is added to the database. The structure of the database is shown below:

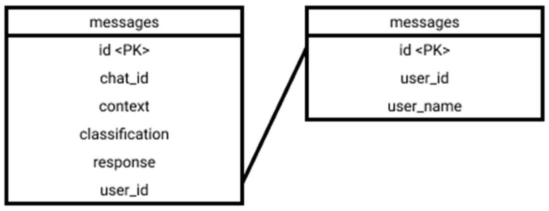

Figure 6: The database schema of the chatbot.

To make the interaction between the chatbot and the database easier, a database manager was created.

III. THE RESULT

Below are shown some examples of the interaction of people with the chatbot:

user: Hi
Bot: hi
user: how are you feeling today?
Bot: fine
user: do you like chocolate?
Bot: yes
user: why do you like chocolate?
Bot: because i m a puppeteer it
user: What do you mean by that?
Bot: i don t know

Now let's see an example of a person that may go through some bad emotions:

Scenario 1:
user: I think that I will end my life
Bot: Do not worry, I am here to help
user: I don't see any sense in living my life
Bot: Don't cry, we're in this together

Scenario 2:
user: I got sick of bullying at school
Bot: Why do you think this is happening?
user: Because my fellows are hating me
Bot: you re not ? ? ?
user: No, I don't hate them
Bot: Remember you're not alone
user: Thank you so much
Bot: you re welcome

IV. CONCLUSION

This work is exploring the implementation in more technical details of a chatbot created to assist people with psychological disorders. The initial problem that this solution is addressing was that people who are struggling with bad feelings or psychological disorders don't always have enough courage to address them to a psychologist. To handle normal dialogs were implemented a Seq2Seq Neural Network base on Loung's attention mechanism, which is generating responses based on the user's input, and a text classification pipeline to identify the emotional state of the person.

Right now the implementation made will never replace a professional psychologist because the models that are used in this chatbot aren't perfect. The text classification model isn't ideal, and a more sensitive model is needed. Also sometimes the text generated by the chatbot isn't always coherent enough. Besides, that in psychology text communication isn't always the unique way to find what is happening with the patient, a lot of information is extracted from non-verbal and para-verbal communication.

However, this experiment offers a new territory to explore, where we can try new models and maybe even new architectures of the chatbot itself.

https://doi.org/10.52326/ic-ecco.2021/CS.06

REFERENCES

[1] Relational Agent for Mental Health | Woebot Health: *https://woebothealth.com/*
[2] Moodnotes: *http://www.thriveport.com/products/moodnotes/*
[3] Wysa: *https://www.wysa.io/*
[4] Youper: *https://www.youper.ai/*
[5] FloydHub's Cornell Movie Corpus preprocessing code: *https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html*
[6] PRAW: The Python Reddit API Wrapper: *https://praw.readthedocs.io/en/stable/*
[7] SMS Spam Collection Dataset: https://www.kaggle.com/uciml/sms-spam-collection-dataset
[8] Official NLTK website: *https://www.nltk.org/*
[9] Ilya Sutskever, Oriol Vinyals, Quoc V. Le. "Sequence to Sequence Learning with Neural Networks": *https://arxiv.org/abs/1409.3215*
[10] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bhdanau, Fethi Bougares, Holgers Schwenk, Yoshua Bengio "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation" https://arxiv.org/abs/1406.1078
[11] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio "Neural Machine Translation by Jointly Learning to Align and Translate" https://arxiv.org/abs/1409.0473
[12] Ming-Thang Loung, Hieu Pham, Christopher D. Manning "Effective Approaches to Attention-based Neural Machine Translation" https://arxiv.org/abs/1508.04025