DEMOKRITOS

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΛΟΠΟΝΝΗΣΟΥ
UNIVERSITY OF
PELOPONNESE

Deep Learning, MSc in Data Science
Assignment 2: Building a ChatBox

VOULGARI ELENI
A.M. 17005

Introduction

The task of this assignment is to build a “chatbox”. A “chatbox” is a computer program or an AI, which conducts a conversation through textual methods. Such models are created to produce real-like simulations of how a human to human verbal interaction is conducted.

For this assignment the Movie dialog corpus from Cornell University is used. The aforementioned dataset includes 220,579 conversational exchanges between 10,292 pairs of movie characters. Even though this is a conversational problem, we can see it as a translational problem with the difference that instead of the target sentences being in another language, them being also in English.

The implemention consists of Recurrent Neural Networks (RNNs) and made of Long-Short Term Memory units (LSTMs cells). RNNs are artificial neural networks, which form a directed graph from the connections between nodes, along a sequence. LSTM cells can conserve the error that is backpropagated through the layers, thus giving a solution to the vanishing/exploding gradient problem.

Data Preprocessing

We will be using the files *movie_lines.txt* and *45000_conversations.txt* (a smaller file of conversations was created, due to the limitation of the hardware).

After exploring the data, some preproccesing must be done in order to help the model to train more efficiently and produce better results. During the cleaning process, two groups of sentences will be created (input and target sentences), all letters will become lowercased, contractions will be replaced, punctuation will be removed and sentences longer than 30 words will not be used.

The general idea is to produce a representation of the input sentences and then map the information, that had been produced, to the target sentences, for the chatbox to train and be able to reply in the user’s question. Therefore, in the feature engineering step, we first create a dictionary with the frequency of the words and remove the rare ones (words that appear less that 5 times) from the inputs and targets. Furthermore, we create dictionaries to provide a unique integer for each word. We also add the unique tokens (<EOS>, <PAD>, <GO>, <UNK>) to the vocabulary dictionaries and create the inversed dictionaries of the aforementioned ones. Finally, we convert the sentences into the number vectors to be fed to the neural network.

Neural Network Architecture

Chatboxes implemented with deep learning techniques are almost all using some variant of the sequence to sequence (Seq2Seq) neural network model. The network is separated into two models. The first model is a RNN called the encoder whereas the second model is also a RNN called the decoder. The encoder takes a sentence as input and processes one word at each timestep. It then converts a sequence of words into a fixed size feature vector, encoding only the important information in the sequence while dropping the irrelevant information.

The output of the encoder can be seen as the summary of the sequence, and it is called the thought vector, because it represents the “intention” of the sequence. This thought vector is taken as an input in the decoder, which generates another sequence, one word at a time. At each timestep,

the output of the decoder is affected by the “thought” of the input and by the previously generated words.

Building the Model

Because of the fact that the seq2seq model cannot handle variable length sequences, we also have to convert the variable length sequences into fixed ones, by padding. We use the aforementioned tokes to fill in the sequence (EOS : End of sentence, PAD : Filler, GO : Start decoding, UNK : Unknown word).

In the process of building the model, we first have to define the input parameters of the sub-models. The input and target placeholders, whose shape is set to [None (batch size), None (length of sentence)], will be fed with their respective input sequence (sentence).

The maximum length of the sentences is not known, so we could find the max length of all sentences and apply it or the max length of the sentences of each batch. Either way, we should add the token <PAD> in the empty positions. Here, we choose to do padding over the maximum size of the sentences of the batch, as we want to minimize the amount of padding, because it is a waste of computational resource. Furthermore, sorting our data by length is a good way to limit padding, although, neural networks may locally overfit to the current size of the sequence.

The next step is to define a way to add the <GO> token in front of all target sentences. This token is used to denote the the start of the decoding.

The encoding layer consists of two sub-layers. The first sub-layer is the embedding layer. The representation of each word in a sentence, will be composed from the features whose number will be specified as a hyperparameter. The second sub-layer is the RNN layers, where the LSTM cells are stacked together, after the dropout technique is being applied.

The decoding phase consists of two separate models, the training decoder and the inference decoder. Among the above decoders, the same embedding vector should be shared. We cannot embed the output from the inference process before running the model, because the output of the current time step will be the input of the next time step, so the training procedure is provided with the embeded input whereas the inference procedure is provided only with the embedding parameters, that are used in the training part.

The inference procedure runs the decoder using just the start-of-sequence token as input and the encoder internal states as the decoder's initial states, then appends the token predicted (after lookup of the token) by the decoder to the decoded sequence and repeats the process with the previously predicted token as input and updates internal states.

The next step is to create the decoding layer. The number of RNN layers in the decoder model has to be equal to the number of RNN layers in the encoder model. In this phase, we create an output layer to map the outputs of the decoder to the tokens of our vocabulary, by a fully connected layer, so as to get the probabilities of occurrence of each word.

Then, we combine together the previously defined, encoding and the decoding layers, to build the seq2seq model. This model defines how the feedforward and backpropagation is executed. We also deside on the optimization algorithm, using Adam optimizer to calculate the

gradient descent on the loss. The loss function used is weighted softmax cross entropy loss function, that is obtained by comparing the predicted values from softmax layer with the target data.

RNNs are known to be affected be the vanishing/exploding gradient, so gradient clipping technique is used to improve the results. Thresholds are used to keep the gradient into a specific boundary [-1,1].

Neural Network Training

Due to the limitations of the hardware (and time) the training of the model was done on a smaller number of conversations (45000). The process took about 5 hours, because due to Resources Exhausted Error, a smaller batch size had to be chosen. The neural network doesn't seem to be learning too much in the 2 epochs we could run.

Results

The model produced the results of the following conversations:

- hello
- you do

- what are you
- i am a <UNK> <EOS>