The Chatbot Assignment

Ilias Zavitsanos

October 11, 2019

1 Introduction

With the advancement of Machine Learning and Deep Learning, machines have started to impersonate as humans. However, the traditional chatbot's dialogue capability is inflexible. It can answer to the user only if there is a pattern-matching between the user query and a set of questions-answers stored in its knowledge base. Deep learning methods, on the other hand, offer the opportunity of new modelling approaches to challenging NLP problems, like the one in this assignment.

The major challenge is to create an adequate sense of context and relate inputs to an output. Several models have been proposed in recent years based on the sequence-to-sequence model in deep recurrent neural networks with attention mechanism, which provides an appropriate architecture to meet this challenge. Related work includes models based on embedding layers, multilayer perceptrons, recurrent neural networks, convolutional neural networks, deep Q networks and so forth.

2 Description

In the context of this assignment, we implement a chatbot based on a neural network architecture, called transformer [1], which is placed amongst the state of the art methods in the field. The transformer is entirely built on the self-attention[1] mechanisms without using sequence-aligned recurrent architecture. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. The major component in the transformer is the unit of the multi-head self-attention mechanism. The transformer views the encoded representation of the input as a set of key-value pairs, (K, V), both of dimension n (input sequence length) in the context of NMT, both the keys and values are the encoder hidden states. In the decoder, the previous output is compressed into a query (Q of dimension m) and the next output is produced by mapping this query and the set of keys and values. The transformer adopts the scaled dot-product attention: the output is a weighted sum of the values, where the weight assigned to each value is determined by the dot-product of the query with all the keys. Rather than only computing the attention once, the multi-head mechanism runs through the scaled dot-product attention multiple times in parallel. The independent attention outputs are simply concatenated and linearly transformed into the expected dimensions.

This architecture makes no assumptions about the temporal or spatial relationships across the data. The layer outputs can be calculated in parallel, distant items can affect each other's output without passing through many recurrent steps or convolution layers and it can learn long-range dependencies. On the other hand, some positional encoding must be added. Otherwise, the model will see a bag of words.

Finally, for fitting the model we calculate the loss based on categorical cross-entropy, since it is the default function to use for multi-class classification problems and in particular we use sparse cross-entropy to save time in memory and computation.

3 Installation and Usage

There are two options for installing and running the project: (a) local installation, and (b) using Docker

[1] An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors.

to build the image. The local installation assumes the existence of Python3 programming language and of a virtual environment with the needed libraries[2]:

- `tensorflow`: used to create, compile and fit the model and then use it for inference
- `tensforflow-datasets`: used to build the input data
- `h5py` and `pyYAML`: used for model persistence
- `emoji`: used for fancy prompt emojis
- `pytest`: used to create and run the test suits

Details on how to create the virtual environment and how to install the requirements are included in the `README.md` file. In this project, we include a trained version of the model, in order to easily test by running `python main.py` inside the `src` folder of the project. You can always get help regarding the usage by typing: `python main.py --help` and you can train the model on whichever dataset you choose by editing the config.ini file and then typing: `python main.py --train`. Trained models are stored inside the models directory of the project. For your convenience, the datasets are included. Details on how to use the model are also included in the `README.md` file.

Given the variety of development environments across users and computers, the application comes Dockerized. If you have `Docker` installed on your machine, all you need to do is build the image and run it, by using the two commands described in the `README.md` file. A Docker container will be created with all the needed libraries and files and a pre-trained model to experiment instantly. Moreover, jupyter notebooks are provided inside the `notebooks` directory.

Finally, unit tests are included. You may need to install pytest in your virtual environment. In order to execute them, just type `pytest` in the root directory of the project.

4 Samples

Model parameters and training time can affect the performance of the model. The `config.ini` file in the root directory contains all the parameters that you can tune in order to achieve different performances than those depicted in the samples below. As a showcase, we include a screenshot depicting that all tests pass and a couple of screenshots with sample utterances of the learned models inside the `samples` folder of the project. Below are a few examples.

```
[Bot] Hello how can I help you?
[User] i need to change my alarms
[Bot] what do you want to change it to?
[User] actually I want to cancel all my
alarms
[Bot] do you want the same alarm settings
for each day?
[User] yes delete them all
[Bot] ok. alarm feed cat has been set to
vibrate.
[User] remove vibrations as well
[Bot] I am happy to help with that
[User] thank you
[Bot] okay, i ve set that alarm to max volume.
there's no way you will sleep through that
[User] no sound at all
[Bot] ok, i ve deleted your alarm
[User] also create a new alarm for Monday
at 7 am
[Bot] ok. i do have a reminder set for am.
delete this alarm. set to vibrate
[User] never mind the reminder.
just 7 am alarm
[Bot] done
```

In the context of this project, we did not find the optimal parameters, nor we explored all the promising data representations. One could experiment with Twitter LDA, which is perfect for short texts, in order to generate topic words for utterances. In that case, we would need to modify the neural network architecture to become "topic-aware" and leverage the topic words obtained from Twitter LDA in question-answer matching.

References

[1] Vaswani et al. Attention is all you need. In Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA, pages 6000–6010.

[2]The required libraries are included in the requirements.txt file of the project