A Chatbot called Alan

Christian Bauer, 01560011

1 Introduction

NLP is a hard task for computers and writing a chatbot that "understands" user input, meaning providing a satisfactory answer is a challenging task. For the project, a chatbot will be implemented, which should be able to answer questions from the *Stanford Question Answering Dataset* (short= *SQuAD*)[1]. This dataset was extracted by volunteers with over 500 Wikipedia articles. Each of these articles is considered a title in the dataset and each holds numerous *question-answer-sets (=QAS)*. This SQuAD-dataset will be the source for the training dataset used to train the machine learning models.

2 Motivation

The usage of neural networks to solve NLP tasks such as translating text with *DeepL* has sparked a great interest for trying to solve a NLP task myself. I decided on implementing a chatbot called *Alan*[2] as the project that will try to answer questions a user can enter via command line.

The "simple" nature of answering questions with available answers is a challenging topic for computers given that the user input question might deter from the question, the model trained with, and therefore, the goal is to dive into approaches to provide a reasonably good model that will provide the correct answer to a proposed question.

The problem is solvable by using *Feed Forward Neural Networks* with hidden layers as well as well-established NLP techniques such as *tokenization, stemming* and *bag of words*.

3 The Learning Task

3.1 Training Experience

The training experience is provided by the University of Stanford in form of QAS. The vast amount of set elements, stated to be over 100.000+ elements of over 500 articles should (presumably) allow the chatbot to cover many topics.

The QAS were provided by volunteers and consist of general questions and answers, as well as unanswerable questions. This is due to the fact, that these unanswerable questions

[1] SQuAD source: https://rajpurkar.github.io/SQuAD-explorer

[2] as homage to Alan Turing

are nonsensical. For example the question "What category of game is Legend of Zelda: Australia Twilight?" is nonsensical because no such game exists. One subtask of this project might be to look into ways to handle this unanswerable questions, by either ignoring them entirely, or finding ways to handle them, like apply an "unanswerable" label to them. Both approaches could be used for the training and then measure the results with the techniques presented in Performance Measure.

3.2 Learning task

For the learning task, NLP techniques such as *tokenization, bag of words* and *stemming* will be used to pre-process the QAS. To process this data, the *Natural Language Toolkit (NLTK)* Python plugin will be used to manipulate the data accordingly.

The dataset itself is stored as a *JSON*-file with a deep structure, that needs to be queried to provide all QAS. From all QAS the questions will be extracted for the training part. The aforementioned NLP-techniques will be used to process the questions in a manner that they can be used as the input signal for the machine learning model.

After the processing of the questions, these will be stored with the corresponding title as the independent-dependent value tuple in a *data loader*.

This data loader will be forwarded into a *Feedforward Neural Network (=FNN)*. The input size will be the size of a *bag of words* list, and the output classes are all possible titles (dependent variable y) that contain the corresponding question-answer sets (dependent variable X). To find a good fitting model, it may be necessary to compare different hidden layer sizes for the FNN. A good fitting model will be determined by using the performance measures described in section 3.3.

3.3 Performance Measure

After the training of the machine learning model, performance measures will be done to be able to test the success of the training. Before diving into the different performance measures that will be used, in the following some commonly used terms will be applied to the *SQuAD*-dataset.

True Positive *The correct answer to a question that is not element of the unanswerable question-answer-subset is called a*

$$\text{True Positive} = T^+.$$

False Positive *If the model provides an answer (that is not the label unanswerable) to an unanswerable question, this will be considered a*

$$\text{False Positive} = F^+.$$

True Negative *If the model classifies a question as unanswerable, and it is indeed an element of the unanswerable set, it is considered a*

$$\text{True Negative} = T^-.$$

False Negative *If the model classifies a question as unanswerable, but this question is not an element of the unanswerable set, it is considered a*

$$\text{False Negative} = F^-.$$

Accuracy [1] *Accuracy is an evaluation metric for classification models. "Informally, accuracy is the fraction of predictions our model got right. [1]"*

$$\text{Accuracy} = \frac{T^+ + T^-}{T^+ + T^- + F^+ + F^-} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Given that numerous unanswerable questions are part of the provided dataset, the accuracy metric might result in a "false sense of confidence" for the machine learning model since accuracy does not take falsely classified questions as much into account as precision does.

Precision [2] *The precision metric is used to show how precise the machine learning model. This is done by calculating how many of the predicted positive values are actually positive values.*

$$\text{Precision} = \frac{T^+}{T^+ + F^+} = \frac{T^+}{\text{Total Predicted Positive}}$$

A precision of over 90% is desirable as the performance goal of the chatbot.

Precision is a good measure if the cost of a F^+ is high. In the case of the *SQuAD*-dataset providing an answer to either an unanswerable question or the wrong answer is considered a F^+. Since both scenarios of F^+ are undesirable and therefore, the precision should be aimed as high as possible, this performance measure will also be the main metric in determining the success rate of a trained model.

Loss Function *The training of the neural network will be done using an optimization process and this requires a loss function to calculate the error the models does while training. After a yet to determine number of training epochs, the calculated loss value should decrease over time. It is desirable to choose a loss function that represents the properties of the problem. There are numerous loss functions available, and the project will most likely implement the*

Cross-Entropy Loss Function.

The reason for this type of loss function is the possibility to calculate the loss for *multi-class classification.*

Weights and Biases *Numerous metrics are available with the usage of the service of Weights and Biases. While training the machine learning model, the process will be logged so different model structures can easily be compared with each other.*

4 Plan

To be able to work comfortably with the dataset, helper classes and functions will be implemented, to easier access elements that are at a deep level in the JSON-file. To be able to use the *bag of words* method, a list of all words will be generated by iterating over all questions available. After storing all possible words in this list, *tokenization* and *stemming* will be used.

Then, a second iteration over the question list will be done to generate the *bag of words* for every question, which will be used as the independent variable (input) x and as the dependent variable, the class variable *title* will be added as y. These values will be stored in a *data loader*, which will be used to train the machine learning model.

For the training of the machine learning model, the values of the *data loader* will be accessed iteratively and feed into the model.

After each training, the machine learning model will be used to calculate the performance measures and these values will be stored locally as well as uploaded to *wandb*.

Given the huge amount of QAS, and transforming each of them into a *bag of words*, it might be necessary to only partially load the data into a *train loader* to be able to train a machine learning model with it on limited hardware resources and once done, load the next sequence into a *train loader*.

5 Related Work

Contextual Chatbots with Tensorflow [3] The website Chatbots Magazine is used as source for the base structure of this chatbot project. In this project a simple chatbot is implemented that receives a small dataset with common greetings, and some further question-answer sets. Other than this project, the framework TensorFlow is used, and the

size of the dataset is rather small. Also, the chatbot only has to identify the correct class of a question, and chooses the answer at random of an arbitrary number of answers. While this approach will be used for this project as well, it might face some issues with choosing the correct answer. Further measures might be necessary to improve the answering of given questions.

Feed-forward neural networks – Why network size is so important The authors of this paper provide methods for finding a good size of FNN.

Given that specific knowledge is not given on my part, the following quote of the paper applies to this project:

> Unfortunately, when no a-priori knowledge about the problem is available, one has to determine the network size by trial and error. Usually, one has to train different size networks and if they don't yield an acceptable solution, then they are discarded. This procedure is repeated until an appropriate network is found [4, Why are small and simple networks better?].

This mentioned steps are the baseline in finding an appropriate model hidden layer size for this project since the input and output size of the network are already defined.

Survey on Chatbot Design Techniques in Speech Conversation Systems [5] The authors of this paper provide a detailed analysis of different chatbot design techniques such as strategies to give reasonable answers to keywords or phrases, and general approaches for implementing a chatbot system.

6 Risk Management

If the chatbot project deems to be unreasonably complicated or outputs nonsense answers, an image classification project using *Convolutional Neural Networks* will be used instead. Multiple online sources [6] [7] [8] are available that describe the problem and some solutions in detail, and therefore, should be feasible to achieve good results.

For this, the package `torchvision` will be heavily used in this case. The necessary GPU power is available in form of an *NVIDIA GeForce 1080TI*.

The mentioned Performance Measure will be used for this alternative option as well.

References

[1] Google, "Classification: Accuracy." Online - developers.google.com, 2020. Available at https://developers.google.com/machine-learning/crash-course/classification/accuracy.

[2] W. Koehrsen, "When accuracy isn't enough, use precision and recall to evaluate your classification model." Online - buildin.com, 2021. Available at https://builtin.com/data-science/precision-and-recall.

[3] G. Kassabgi, "Contextual chatbots with tensorflow." Chatbots Magazine, 2017.

[4] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *IEEE Potentials*, vol. 13, no. 4, pp. 27–31, 1994.

[5] S. A. Abdul-Kader and J. C. Woods, "Survey on chatbot design techniques in speech conversation systems," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 7, 2015.

[6] A. Rosebrock, "Pytorch: Training your first convolutional neural network." Online - pyimagenet, 2021. Available at https://www.pyimagesearch.com/2021/07/19/pytorch-training-your-first-convolutional-neural-network-cnn.

[7] A. Rosebrock, "Pytorch image classification with pre-trained networks." Online - pyimagenet, 2021. Available at https://www.pyimagesearch.com/2021/07/26/pytorch-image-classification-with-pre-trained-networks.

[8] PyTorch, "Training a classifier." Online - pytorch.org, 2021. Available at https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html.