Healthcare Support Chatbot

Link to GitHub Repository

Link to Video Presentation

1 Introduction

Chatbots are artificial intelligence (AI) software programs designed with machine learning algorithms, including natural language processing (NLP) [1], [2]. In healthcare, chatbots, also known as healthbots, provide support to healthcare workers. They introduce solutions to problems such as the high workload of medical staff, unnecessary treatments and procedures, consultation times, and hospital readmissions [2].

Chatbots can be designed to provide medical and diagnostic information in response to early signs of illness. They can also connect people to the right healthcare providers in the community. Because they are AI, they are not biased towards certain user demographics and are better at maintaining confidentiality. Chatbots are always available as they do not fatigue or get sick and are only disturbed by network faults [2]. This also makes them very cost effective, as the only cost is in the initial creation and training of the chatbot, and occasional maintenance.

These reasons are our motivation for creating our healthcare support chatbot for first-response interactions with the capability of predicting a diagnosis. This project required two algorithms for two separate tasks: intent classification and disease prediction. In this report we implement and compare the performance of the two versions using different models.

- The input to our chatbot Sam is a pre-processed user input string of variable length. We then use a fully connected feed-forward neural network to output the predicted intent of the user. The output is the index of the predicted user intent.
- The input to our chatbot Ayla is a pre-processed user input string of variable length. We then use a feed-forward neural network built on top of a pretrained DistilBERT model to output the predicted intent of the user. The output is the index of the predicted user intent.
- The input to our disease prediction algorithm is a stored and pre-processed list of symptoms of variable length. We then use a fully connected feed-forward neural network to predict a single index which corresponds to a disease.

2 Related works

2.1 Feed-forward neural networks

Feed-forward neural networks are simple. They train very quickly and can respond very quickly when implemented. One paper [3] used NLP pre-processing from the 'nltk' toolkit. In that study, the input query was tokenized and stemmed then encoded using the bag of words. This input was passed through the feed-forward neural network which returned an 'intent'. This intent identification was then used to retrieve a corresponding random reply from a predetermined pool of responses. The final chatbot was presented in a GUI made with the python 'Tkinter' package. We used a similar

Team Members: Fatemeh (Ava) Khosravi, Lachlan Muirden, Zaria Imran

approach to train Sam and our disease classifier due to its simplicity, and we also used the 'Tkinter' package for our GUI.

2.2 Bidirectional Recurrent Neural Networks

Bidirectional recurrent neural networks are a popular option for retrieval-based chatbots [4], [5], [6]. The innovation Moore et al. [4] have made in their LSTM model is a hierarchical system with 3 modules: word embedding, utterance embedding and context embedding. The advantage of this structure is that it learns a larger representation of each dialogue at the 3 levels. A similar approach is done in another paper [5] where the value of context is "the weight of features from different timestamp[s]". One paper [6] uses beam search decoding to build a state-space tree. They used greedy search to identify the most likely next state. This reduced the number of options for "reasonable next steps" based on heuristic values. If no response was found to the query in the expanded state, the beam would be expanded further. Our team used a pretrained recurrent neural network (DistilBERT).

2.3 Restricted Boltzmann Machine (RBM)

RBM is a generative algorithm which predicts the input from the model output to train itself. Argal et al. [7] proposed RBM for collaborative filtering. In their paper, the chatbot prompted the user to fill the gaps and provided a prediction based on the information and history provided. The existing collaborative filtering would not work with large databases so RBMs were used to model data like user ratings of different objects. The method involved creating RBMs equal to the number of users then treating each as an individual training set. The softmax units were connected to binary hidden units in a symmetric configuration which allowed the learning to be for the dependencies between ratings of each user. The weight and bias were shared between the user's softmax unit and the hidden layers. The resulting chatbot could provide recommendations and remember user slot values using previous conversations in the same session. This type of algorithm seems like it would be useful for our purposes in disease prediction where symptoms may be predicted and confirmed with the user in order to make a better prediction.

2.4 Subword Semantic Hashing

Subword semantic hashing is a method of tokenizing which is "accurate, versatile, stable, and fast" [8]. The words are passed through a pre-hashing function which appends a hash to the start and end of the word and returns trigram sub tokens. Our team did not use this type of tokenizing/stemming but found it was an interesting and simple approach with significant results.

2.5 State of the Art Pretraining

Pretraining of models is performed on large-scale unlabelled text databases. In 2020, the Google AI Brain Team [9] released the XLNet with a generalised autoregression pretraining method. In the paper, they analyse BERT and AR Conventional Modelling. BERT denoises auto-encoding by corrupting text input via masking and reconstructing the masked inputs from the non-masked inputs. AR conventional modelling is done by maximising the likelihood under forward autoregressive factorization. XLNet aims to achieve a mix of both models' advantages: the AR conventional model's universal product rule and non-reliance on input corruption but also BERT's bidirectional context dependency.

The XLNet paper proposed permutation language modelling as a pretraining objective. It has two-stream self-attention where the overall goal is to collect contextual information from the target position. The paper does this by having two sets of representation: content, which encodes both context and input, and query, which has access to context and target position. They incorporate ideas from the previous state-of-the-art Transformer-XL: relative positional encoding scheme and segment recurrence mechanism. They only perform partial prediction on a subset of tokens to optimise training. This saves speed and memory.

3 IMPLEMENTATION

3.1 DATASET

Link to Data File Source

3.1.1 Description of Data Files

1. Dataset.csv
2. Symptom_description.csv
3. Symptom_precaution.csv

Data file 1 consists of 18 columns and 4920 rows. Each row contains a disease name and a maximum of 17 corresponding symptoms. There are many repeats of each disease with some having different combinations of symptoms. Data file 2 has 41 rows referring to the disease number with 2 columns being the disease name and its corresponding disease description. Data file 3 consists of the number of diseases as its rows and the columns include the disease name and 4 disease precautions each resulting in 41 rows and 5 columns.

3.1.2 Cleaning the Dataset

From the first data file, row 4879 to row 4922 were used for training the disease classifier (training dataset size: 41) and the rest was used for testing (testing dataset size: 4879).

The dataset required for training the chatbot intent classifier was created by our team using data files 2 and 3. Each intent label has a corresponding set of patterns, responses, and context as shown in the table below. The resultant dataset contained 1320 entries corresponding to 263 unique labels. Our team randomly selected two percent of intents.json for the testing set which left 98% for training.

intents.json	**Purpose**
Labels	Target output of model
Patterns	Inputs for model
Responses	Predefined responses
Context	Input for response choice

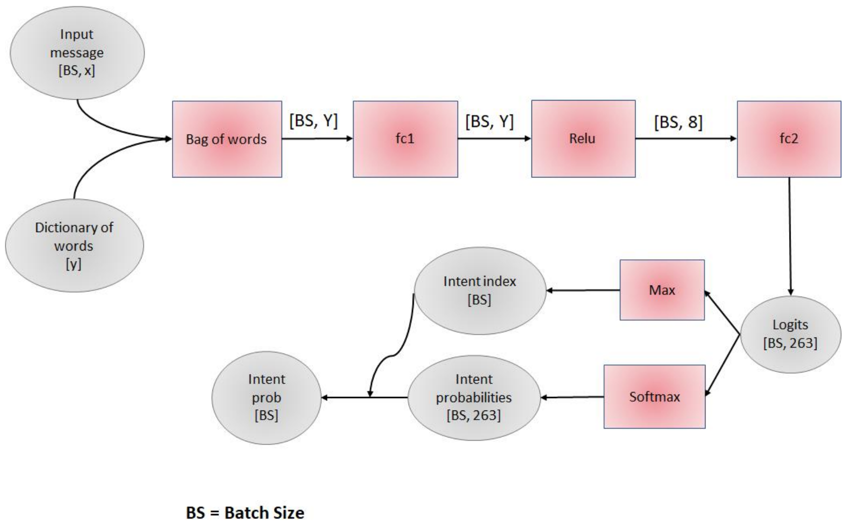
An example of a created 'intent' can be found in the Appendix.

3.2 SAM: LINEAR MODEL FOR INTENT CLASSIFICATION

3.2.1 Model Implementation

A simple feed-forward network was used for this model. This network uses inputs that are encoded by basic natural language processing techniques (implementations of these in the Appendix).

A flow diagram of the linear model is shown below:



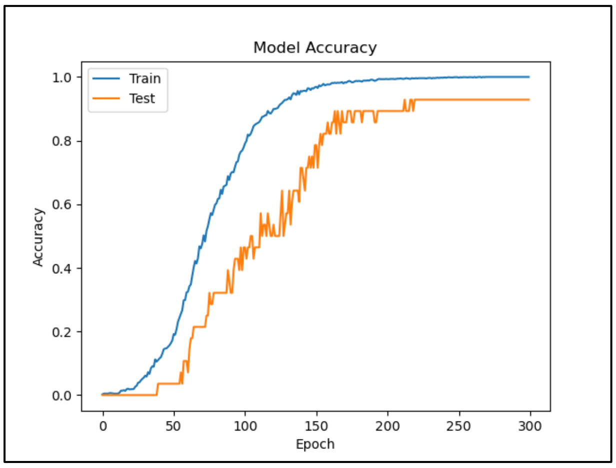
The final output label (integer between 0 and 263) represents the identified intent of the user which was determined based on the user input message. The correlating probability is also output by the model.

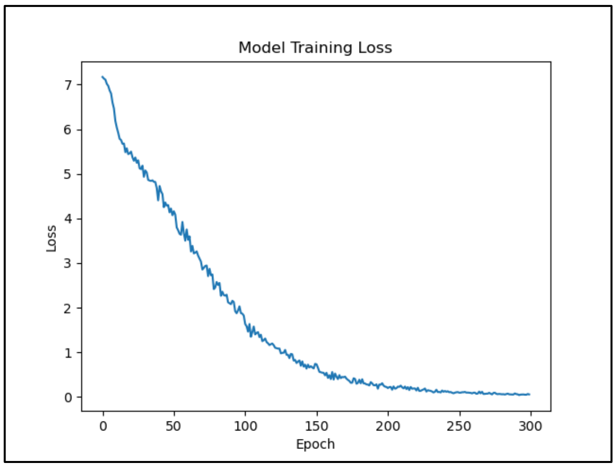
3.2.2 Hyperparameters

Number of epochs	300
Batch size	100
Learning rate	0.001
Loss function	Cross Entropy Loss
Optimizer	Adam

The learning rate choice was fundamental for training Sam. A learning rate larger than 0.001 resulted in an increase in loss during training which indicates that the model was overshooting the minima. Cross entropy loss is the default selection for multi-class classification, see equation in Appendix.

3.2.3 Evaluation: Accuracy and Loss





The linear model training was fast at 0.47 seconds with a i7 1.80 GHz CPU and converged after 250 epochs. The training results converged quickly but the accuracy and loss curves were noisy which suggests a slightly smaller learning rate may have benefitted training.

3.3 AYLA: BERT MODEL FOR INTENT CLASSIFICATION

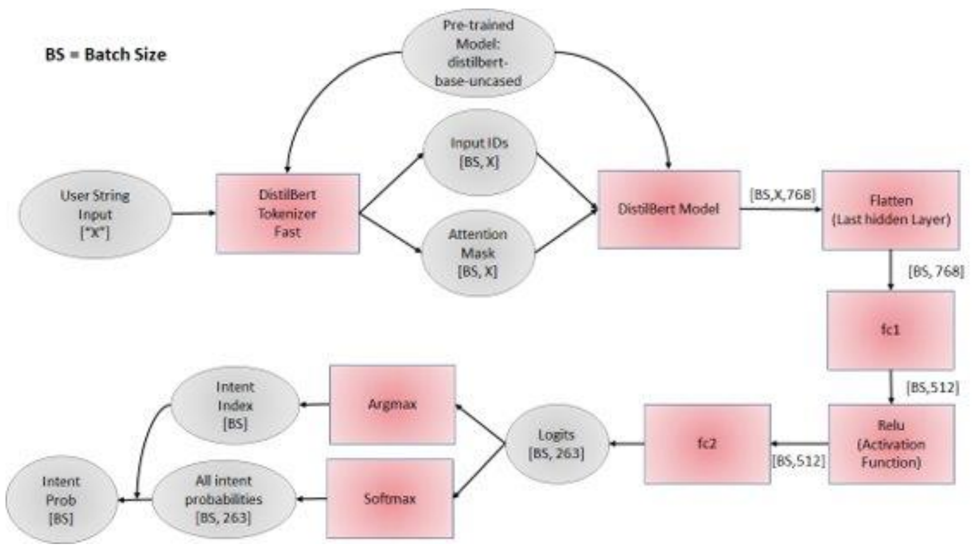
3.3.1 Model Implementation

Link for downloading trained Ayla

For this model we used a pretrained network – the distilled version of the original BERT model (base 12 Encoders with 12 bidirectional self-attention heads). Distilling a model is the process of teaching a smaller network (DistilBERT) to behave like the larger network (BERT) with an increased efficiency. The model we used had been trained for 90 hours on 8 16GB V100 [10]. The DistilBERT base model used is uncased – this refers to not distinguishing between upper and lower cases of characters. The DistilBERT tokenizer was also loaded from the same pretrained model – the code for this can be found in the Appendix. The DistilBERT model was chosen over the XLNet due to its size – it is smaller and easier to train than XLNet which is seen in Section 2.5.

Link to DistilBERT base model uncased

A flow diagram of the DistilBERT model with the linear classifier is below:



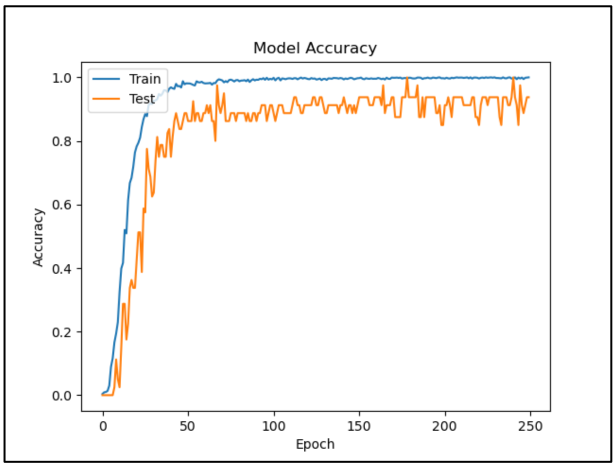
The final output label (integer between 0 and 263) represents the identified intent of the user which was determined based on user input message. The correlating probability is also output by the model.

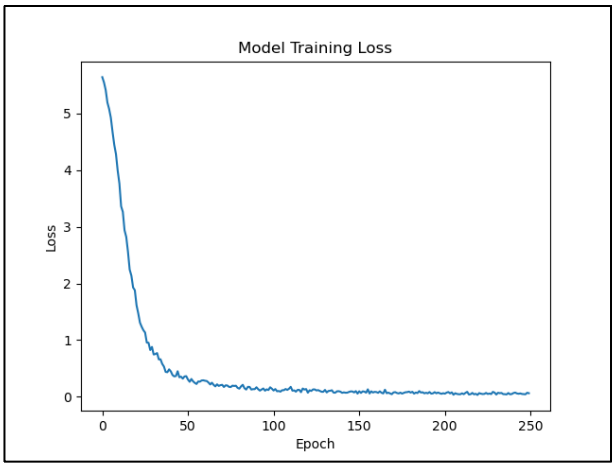
3.3.2 Hyperparameters

Number of epochs	250
Batch size	500
Learning rate	0.01
Loss function	Cross Entropy Loss
Optimizer	AdamW

The most important parameter for our training of Ayla was the optimizer. When using Adam, the training did not improve despite trying many different learning rates. AdamW instead incorporates a weight decay component into the loss and was the key factor in being able to train our model. A weight decay optimiser works by adding a loss component equal to the squared sum of the weights multiplied by a weight decay constant. A weight decay optimiser works by adding a loss component correlated to the magnitude of the weights. This way, if a weight is not contributing sufficiently to decreasing the loss, the additional loss contributed by the weight decay component will cause the magnitude of the weight to be reduced. Networks with smaller weights overfit less and generalize better which was important for this large network [11]. The other highly important parameter for training Ayla was the learning rate choice, as larger than 0.01 caused instability of the loss rather than the steady decrease seen at a learning rate of 0.01. Cross entropy loss is the default selection for multi-class classification, see equation in the Appendix.

3.3.3 Evaluation: Accuracy and Loss



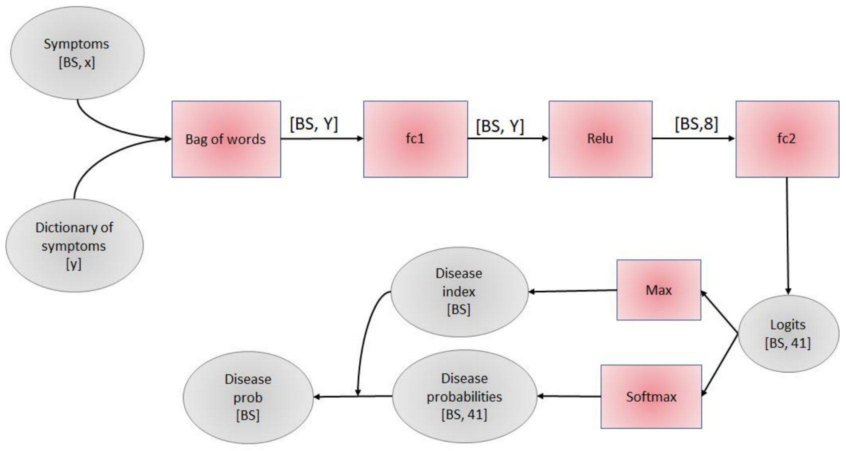


The training of the Bert model with a linear classifier converged after 100 epochs. The time taken to train per epoch was 23.83 seconds when training with a 4GB GeForce GTX1050 Ti with Max-Q. Ayla took a significantly longer time to train compared to Sam which was expected as the model was training on data being output by a complex model. After being passed through the DistilBERT model the linear classifier was learning to interpret the output from the last hidden layer (768 units) and predict an intent.

3.4 Linear Model for Disease Classification

3.4.1 Model Implementation

A flow diagram of the linear model is below:



BS = Batch Size

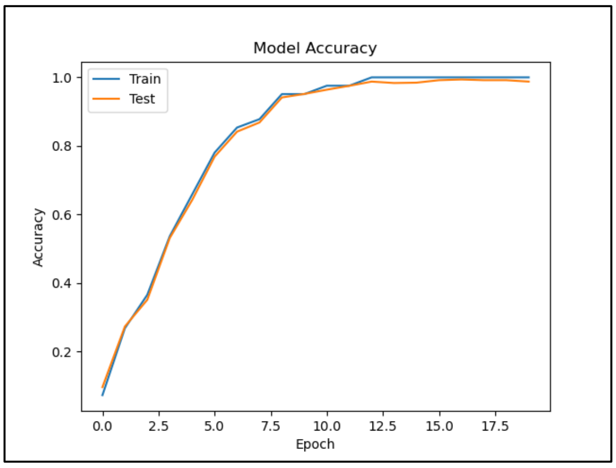
The final output label (integer between 0 and 41) represents the identified disease of the user which was determined based on the stored symptoms collected during conversation. The correlating probability is also output by the model.

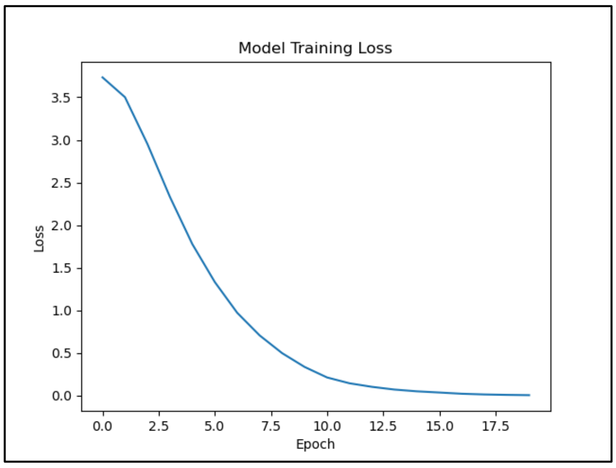
3.4.2 Hyperparameters

Number of epochs	20
Batch size	41
Learning rate	0.1
Loss function	Cross Entropy Loss
Optimizer	Adam

The batch size for this training is the same size as the dataset being used – which meant every epoch was updating ideally for every disease. Due to the small amount of data, training in this way was possible. Cross entropy loss is the default selection for multi-class classification, see equation in Appendix.

3.4.3 Evaluation: Accuracy and Loss



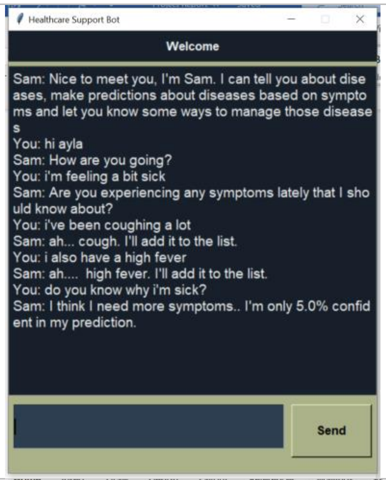
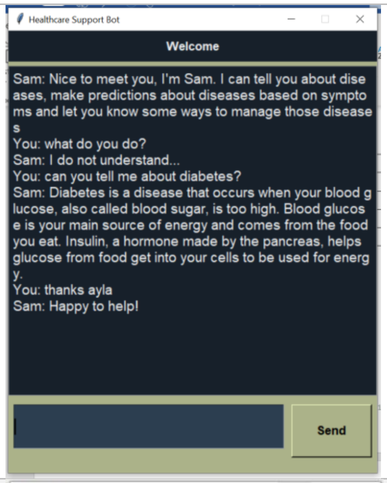
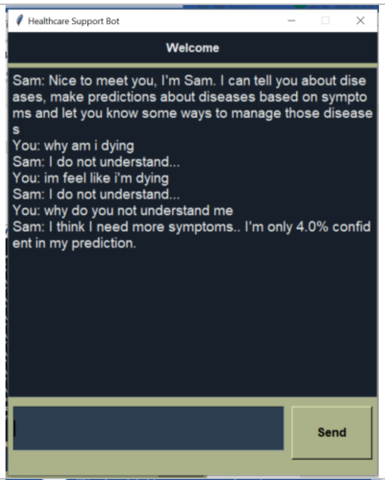
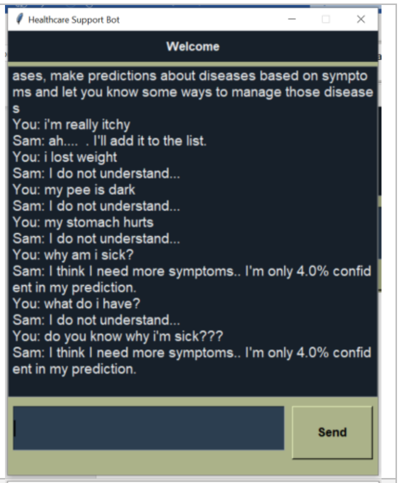
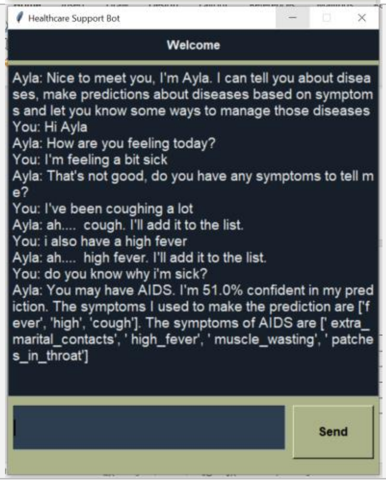
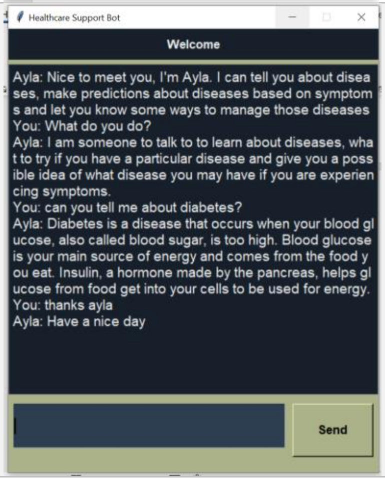
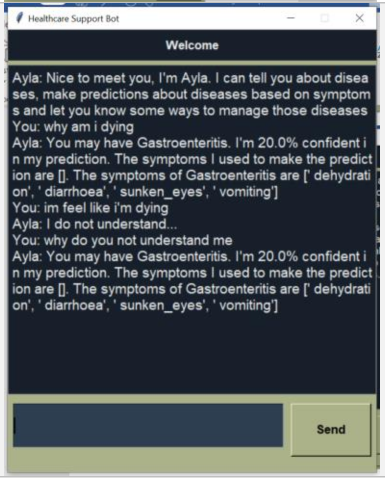
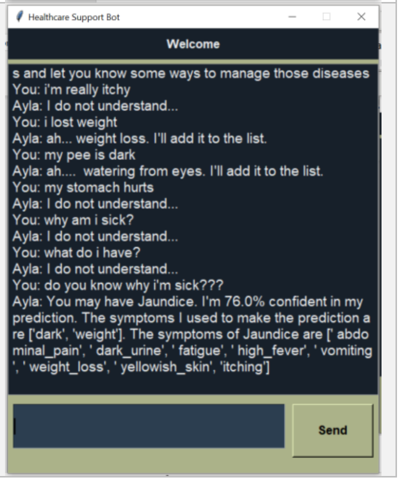


The training for the disease classifier converged after 15 epochs and trained in <0.2 seconds per epoch with an i7 1.80 GHz CPU. The training loss curve shown above is smooth. The model reaches near 100% accuracy on both training and test sets.

Team Members: Fatemeh (Ava) Khosravi, Lachlan Muirden, Zaria Imran

3.5 Chatbot Performance of Linear vs Bert Model: Examples of Conversation

The GUI for our chatbot was created using the Tkinter package due to its simplicity of implementation. The disease classifier was trained separately to the chatbots but as can be seen from the different responses – the output of the disease classifier is not consistent for every training run.

Linear Model	 Healthcare Support Bot Welcome Sam: Nice to meet you, I'm Sam. I can tell you about dise ases, make predictions about diseases based on sympto ms and let you know some ways to manage those disease s You: hi ayla Sam: How are you going? You: i'm feeling a bit sick Sam: Are you experiencing any symptoms lately that I sho uld know about? You: i've been coughing a lot Sam: ah... cough. I'll add it to the list. You: i also have a high fever Sam: ah.... high fever. I'll add it to the list. You: do you know why i'm sick? Sam: I think I need more symptoms.. I'm only 5.0% confid ent in my prediction. Send	 Healthcare Support Bot Welcome Sam: Nice to meet you, I'm Sam. I can tell you about dise ases, make predictions about diseases based on sympto ms and let you know some ways to manage those disease s You: what do you do? Sam: I do not understand... You: can you tell me about diabetes? Sam: Diabetes is a disease that occurs when your blood g lucose, also called blood sugar, is too high. Blood glucos e is your main source of energy and comes from the food you eat. Insulin, a hormone made by the pancreas, helps glucose from food get into your cells to be used for energ y. You: thanks ayla Sam: Happy to help! Send	 Healthcare Support Bot Welcome Sam: Nice to meet you, I'm Sam. I can tell you about dise ases, make predictions about diseases based on sympto ms and let you know some ways to manage those disease s You: why am i dying Sam: I do not understand... You: im feel like i'm dying Sam: I do not understand... You: why do you not understand me Sam: I think I need more symptoms.. I'm only 4.0% confid ent in my prediction. Send	 Healthcare Support Bot Welcome ases, make predictions about diseases based on sympto ms and let you know some ways to manage those disease s You: i'm really itchy Sam: ah.... . I'll add it to the list. You: i lost weight Sam: I do not understand... You: my pee is dark Sam: I do not understand... You: my stomach hurts Sam: I do not understand... You: why am i sick? Sam: I think I need more symptoms.. I'm only 4.0% confid ent in my prediction. You: what do i have? Sam: I do not understand... You: do you know why i'm sick??? Sam: I think I need more symptoms.. I'm only 4.0% confid ent in my prediction. Send
DistilBERT Model	 Healthcare Support Bot Welcome Ayla: Nice to meet you, I'm Ayla. I can tell you about disea ses, make predictions about diseases based on symptom s and let you know some ways to manage those diseases You: Hi Ayla Ayla: How are you feeling today? You: I'm feeling a bit sick Ayla: That's not good, do you have any symptoms to tell m e? You: I've been coughing a lot Ayla: ah.... cough. I'll add it to the list. You: i also have a high fever Ayla: ah.... high fever. I'll add it to the list. You: do you know why i'm sick? Ayla: You may have AIDS. I'm 51.0% confident in my pred iction. The symptoms I used to make the prediction are ['f ever', 'high', 'cough']. The symptoms of AIDS are [' extra_ marital_contacts', ' high_fever', ' muscle_wasting', ' patche s_in_throat'] Send	 Healthcare Support Bot Welcome Ayla: Nice to meet you, I'm Ayla. I can tell you about disea ses, make predictions about diseases based on symptom s and let you know some ways to manage those diseases You: What do you do? Ayla: I am someone to talk to to learn about diseases, wha t to try if you have a particular disease and give you a poss ible idea of what disease you may have if you are experien cing symptoms. You: can you tell me about diabetes? Ayla: Diabetes is a disease that occurs when your blood gl ucose, also called blood sugar, is too high. Blood glucose is your main source of energy and comes from the food y ou eat. Insulin, a hormone made by the pancreas, helps gl ucose from food get into your cells to be used for energy. You: thanks ayla Ayla: Have a nice day Send	 Healthcare Support Bot Welcome Ayla: Nice to meet you, I'm Ayla. I can tell you about disea ses, make predictions about diseases based on symptom s and let you know some ways to manage those diseases You: why am i dying Ayla: You may have Gastroenteritis. I'm 20.0% confident i n my prediction. The symptoms I used to make the predict ion are []. The symptoms of Gastroenteritis are [' dehydrati on', ' diarrhoea', ' sunken_eyes', ' vomiting'] You: im feel like i'm dying Ayla: I do not understand... You: why do you not understand me Ayla: You may have Gastroenteritis. I'm 20.0% confident i n my prediction. The symptoms I used to make the predict ion are []. The symptoms of Gastroenteritis are [' dehydrati on', ' diarrhoea', ' sunken_eyes', ' vomiting'] Send	 Healthcare Support Bot Welcome s and let you know some ways to manage those diseases You: i'm really itchy Ayla: I do not understand... You: i lost weight Ayla: ah... weight loss. I'll add it to the list. You: my pee is dark Ayla: ah.... watering from eyes. I'll add it to the list. You: my stomach hurts Ayla: I do not understand... You: why am i sick? Ayla: I do not understand... You: what do i have? Ayla: I do not understand... You: do you know why i'm sick??? Ayla: You may have Jaundice. I'm 76.0% confident in my prediction. The symptoms I used to make the prediction a re ['dark', 'weight']. The symptoms of Jaundice are [' abdo minal_pain', ' dark_urine', ' fatigue', ' high_fever', ' vomiting ', ' weight_loss', ' yellowish_skin', 'itching'] Send

Team Members: Fatemeh (Ava) Khosravi, Lachlan Muirden, Zaria Imran

4 Conclusions

At the completion of this project, the chatbots Sam and Ayla were trained and capable of responding. However, the sensibility of their responses was limited at times as seen in Section 3.5. The key factor that resulted in the failures of our chatbot was the lack of a good dataset. Due to the dataset being created by our team as described in Section 3.1.2 it has a significant limitation: the models only learn limited patterns and are limited by how comprehensive we were and our own vocabulary. With a more suitable dataset, the DistilBERT model Ayla should have outperformed Sam's linear model.

The disease classification model seemed to train very well however did not produce consistent responses. The model therefore makes connections that cannot be generalised due to such a small training set with no variation per label. This is especially true in the case where Ayla is 20% confident the user has Gastroenteritis despite being given no symptoms. When predicting disease in practice there is a limited amount of information (number of symptoms provided) which is not the case for the testing which is why testing set accuracy is so different to the sensibility of the chatbot.

4.1 Future Development

The time to respond is a significant factor in choosing a model. The relatively instant wait time for the linear model requires delay to be perceived as a thoughtful response. The response from the DistilBERT model takes longer but this could be rectified from user perspective with a "typing" signal like Facebook messenger.

The disease classification model could be improved by adapting to the RBM from Section 2.3. With RBM the model could identify its gaps and either prompt the user or fill it in based on probability, which would be a significant improvement.

Improvements to the dataset could be done in many ways. To use the DistilBERT model effectively, training on a large dataset of collected GP interactions would be ideal. Implementing the ability to identify synonyms would also significantly benefit the model. Synonyms could be represented by the same token, for example.

If the intents.json file were still to be used as the dataset, reducing the number of intents would benefit output of the chat model. In this case, talking about symptoms would be classed as one intent rather than a separate intent for each symptom as we have done. Dealing with the input message would be done interactively – searching the csv directly for a match. This would both increase sensibility of the output and make it easier to expand the capabilities of the chatbot if new datasets were added. Ideally, beam search decoding would also be implemented as described in Section 2.2, as the purpose of the healthcare chatbot would follow a common pattern that can be learnt.

A significant addition to the capabilities of our chatbot would be incorporating sensor data – especially those found on a mobile phone directly or in attachments. The chatbot would be an app on the phone and the sensors would be able to detect temperature, shakiness or take photos and detect symptoms based on these observations. This would provide a first response clinical experience which is the purpose of the healthbot.

REFERENCES

[1] A. Palanica, P. Flaschner, A. Thommandram, M. Li and Y. Fossat, “Physicians’ Perceptions of Chatbots in Health Care: Cross-Sectional Web-Based Survey”, J Med Internet Res vol. 21 no. 4, Nov. 2019. doi: 10.2196/12887

[2] K. Kalinin. “Medical Chatbots: The Future of the Healthcare Industry.” https://topflightapps.com/ideas/chatbots-in-healthcare/ (Accessed: Oct. 5, 2021)

[3] P. Kandpal, K. Jasnani, R. Raut and S. Bhorge, "Contextual Chatbot for Healthcare Purposes (using Deep Learning)," 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), 2020, pp. 625-634, doi: 10.1109/WorldS450073.2020.9210351.

[4] K. Moore et al, “A comprehensive solution to retrieval-based chatbot construction”, arXiv:2106.06139v1, Jun. 2021

[5] M. Dhyani and R. Kumar, “An intelligent Chatbot using deep learning with Bidirectional RNN and attention model”, Mater Today Proc. vol. 34, pp. 817-824, Jun. 2020. doi:10.1016/j.matpr.2020.05.450

[6] S. P. Reddy Karri and B. Santhosh Kumar, "Deep Learning Techniques for Implementation of Chatbots," 2020 International Conference on Computer Communication and Informatics (ICCCI), 2020, pp. 1-5, doi: 10.1109/ICCCI48352.2020.9104143.

[7] A. Argal, S. Gupta, A. Modi, P. Pandey, S. Shim and C. Choo, "Intelligent travel chatbot for predictive recommendation in echo platform," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), 2018, pp. 176-183, doi: 10.1109/CCWC.2018.8301732.

[8] K. Shridar et al, “Subword Semantic Hashing for Intent Classification on Small Datasets”, arXiv:1810.07150v3, Sep. 2019, doi: 10.1109/IJCNN.2019.8852420

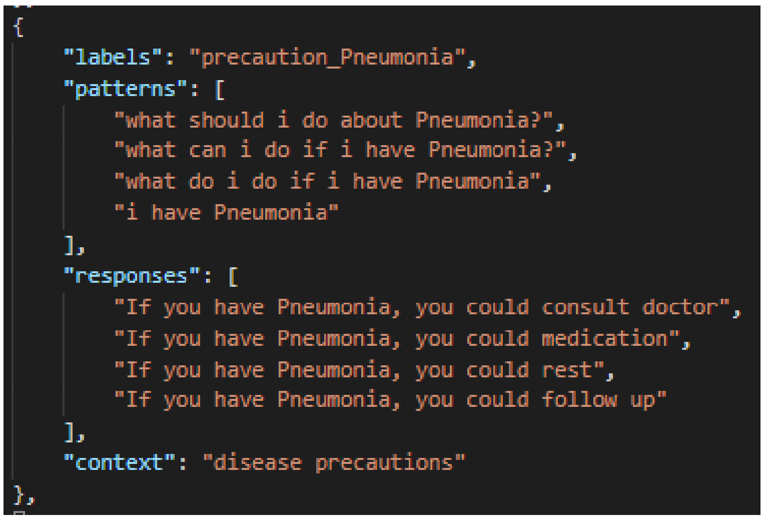
[9] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov and Q. Le, “XLNet: Generalized Autoregressive Pretraining for Language Understanding”, arXiv:1906.08237v2, Jan. 2020

[10] V. Sanh, L. Debut, J. Chaumond and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”, arXiv:1910.01108v4, Mar. 2020

[11] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization”, arXiv:1711.05101v3, Jan. 2019

APPENDIX

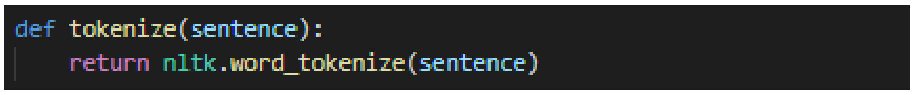
INTENTS.JSON: EXAMPLE



```
{
    "labels": "precaution_Pneumonia",
    "patterns": [
        "what should i do about Pneumonia?",
        "what can i do if i have Pneumonia?",
        "what do i do if i have Pneumonia",
        "i have Pneumonia"
    ],
    "responses": [
        "If you have Pneumonia, you could consult doctor",
        "If you have Pneumonia, you could medication",
        "If you have Pneumonia, you could rest",
        "If you have Pneumonia, you could follow up"
    ],
    "context": "disease precautions"
},
```

NLP 'NLTK' IMPLEMENTATIONS

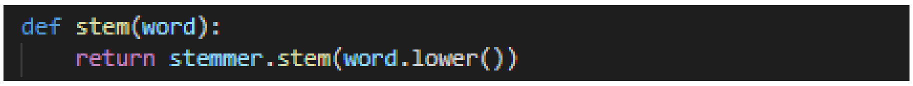
Tokenizing



```python
def tokenize(sentence):
    return nltk.word_tokenize(sentence)
```

Tokenizing is the process of splitting a sentence up into its words.
e.g. string input "I am Sam" -> list of strings outputs ["I", "am", "Sam"]

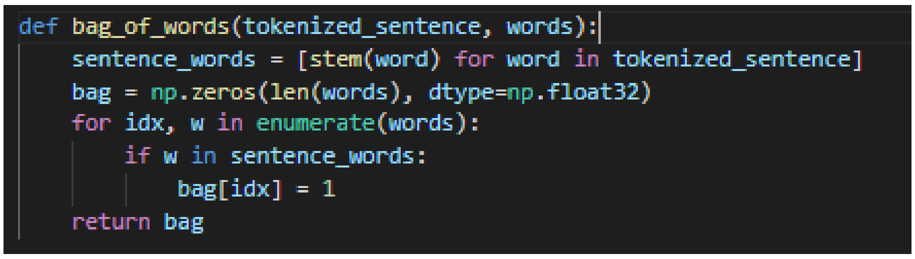
Stemming



```python
def stem(word):
    return stemmer.stem(word.lower())
```

Stemming is the process of retrieving the root of a word.
e.g. word "colder" -> root word "cold"

Bag of Words



```python
def bag_of_words(tokenized_sentence, words):
    sentence_words = [stem(word) for word in tokenized_sentence]
    bag = np.zeros(len(words), dtype=np.float32)
    for idx, w in enumerate(words):
        if w in sentence_words:
            bag[idx] = 1
    return bag
```

The bag of words is the method of encoding the strings into a feature space. The bag of words will contain all stemmed possible words collected from the training dataset. The output of the bag of words function will be an array of frequencies of each word in the stemmed input sentence.

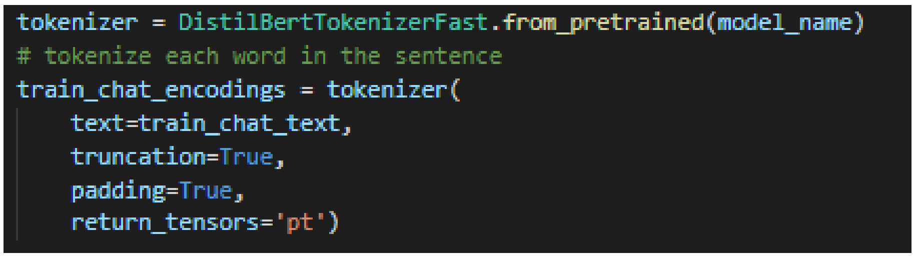
e.g. input [“I”, “am”, “cold”, “to”] + dictionary [“I”, “not”, “you”, “cold”, “am”, “to”] -> encoded input [1,0,0,1,1,1]

Cross Entropy Loss Equation

Cross entropy loss is calculated using the following equation. The letter ‘m’ represents the number of data pairs whilst ‘y_i’ and ‘$\bar{y}_i$’ represent the output and target output respectively. This is the default choice for multi-class classification because it allows each datapoint to be given an associated probability, which can then be compared, and the class with the greatest associated probability can be assigned as the correct class.

$$Cross\ Entropy\ Loss = -\frac{1}{m}\sum_{i=1}^{m} y_i \cdot \log(\widehat{y_i})$$

DistilBERT Tokenizer



```
tokenizer = DistilBertTokenizerFast.from_pretrained(model_name)
# tokenize each word in the sentence
train_chat_encodings = tokenizer(
    text=train_chat_text,
    truncation=True,
    padding=True,
    return_tensors='pt')
```

The DistilBERT tokenizer takes input text in the form of a string, and outputs that same string as a vector. This vector is defined by the dictionary of the pretrained model, which in this case is distilBERT-base-uncased (link). The tokenizer also outputs an attention vector which indicated how many words were in the input message. This allows the DistilBERT model to take any length of input. Truncation and padding are added to either cut off or extend the output to a certain length, and the tensors are returned as pytorch tensors specifically.