Conversational AI Chatbot
Interim Report

Author
Fady Benattayallah
Student ID: 14307825
psyfb2@nottingham.ac.uk

Supervisor
Ke Zhou
pszkz@exmail.nottingham.ac.uk

6th December 2019

Abstract

Conversational artificial intelligence is still considered to be in its infancy with many companies and universities investing heavily in this area of AI. Although great progress has already been made with commercial social bots such as Siri and Amazon Alexa, no system is close to passing the turing test. This is because conversational chat-bots suffer from a number of issues. One of them being the inability to keep a consistent persona and set of backgrounds facts.

For example, a social-bot is likely to give contradicting answers to the same question asked twice, if the question is paraphrased the second time. The main goal is to minimize this behaviour as much as possible by using machine learning models which are variations of the Recurrent Neural Network (RNN).

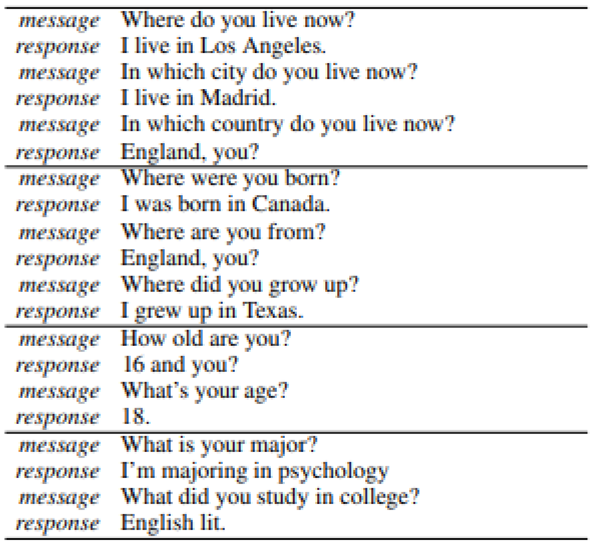
Contents

1 Introduction and Motivation

Conversational AI has advanced significantly in recent years with the rise of virtual assistants such as Siri and Amazon Alexa. Amazon has invested heavily into the advancement of conversational AI with the establishment of the Alex Prize [1]. A competition where research groups are given a grant of $250,000 to create a realistic social-bot which is capable of outperforming Alexa in its current state with the top 3 groups winning a larger grant and inclusion of their work into the next version of Alexa.
However conversational AI is still in it's infancy and there are three main problems which social-bots face today:

- Lack of consistent persona [6]. This means social-bots do not maintain a set of background facts about themselves and are likely to contradict themselves even within a single short conversation.
- Lack of long term memory with respect to the current conversation as models are usually trained to produce an utterance based mostly on the recent dialogue [15]. This means the social-bot is likely to forget information earlier in the conversation leading to unnatural interactions with humans.
- Predisposition to answer questions with non-specific answers such as "I'm not sure" [5].

Each of these individually are hard to solve problems in their own respect. This project will be focused on solving the problem of a lack of persona. The overall goal is to create a social-bot which can participate in a non-goal oriented dialogue while maintaining a consistent persona. Concretely, a persona can be defined as a set of background facts which are remembered and used during conversation. When asking a chatbot multiple paraphrased questions the responses may show a lack of consistency. This occurs because the social-bot will reflect the persona of the responses within the dataset, but these responses come from many different people with different personalities and background facts.



message	Where do you live now?
response	I live in Los Angeles.
message	In which city do you live now?
response	I live in Madrid.
message	In which country do you live now?
response	England, you?
message	Where were you born?
response	I was born in Canada.
message	Where are you from?
response	England, you?
message	Where did you grow up?
response	I grew up in Texas.
message	How old are you?
response	16 and you?
message	What's your age?
response	18.
message	What is your major?
response	I'm majoring in psychology
message	What did you study in college?
response	English lit.

Figure 1: Example of inconsistent persona generated by a "4-layer SEQ2SEQ model trained on 25 million Twitter conversation snippets" [6].

The lack of a consistent and coherent persona makes it impossible for social-bots in there current state to pass the turing test because some of the answers the social-bot gives will be obvious contradictions to previous answers making the distinction between human and chat-bot clear. The key objectives of this project are to:

- Implement and train a Sequence to Sequence neural network architecture [13] to allow the social-bot to map input questions of varying lengths to meaningful answers as a base model.
- Implement the Speaker Model [6] on top of this Sequence to Sequence base model in order to introduce a consistent persona which the Sequence to Sequence model will lack.
- Implement Attention [2] on top of the Speaker Model.
- Use Glove word embeddings [10] in order to convert words into fixed length vectors.
- Investigate other models such as the Generative Modeling with Sparse Transformers [14].
- Compare the effectiveness of the models trained by using automatic and human evaluation metrics.

2 Methodology

2.1 Summary of Problem

The objective is to create a non-goal oriented chit-chat dialogue system. This is a supervised learning problem where each training example is a response to an utterance. The words in the input sequence are the features and the words in the response sequence are the labels. Our model should learn to output an appropriate reply to an input utterance with consistent persona by using a large text corpus of conversations.

Deep Artificial Neural Networks (DNN) [12] are very powerful and have been used to solve a huge range of problems successfully, especially for classification problems for example, image recognition or spam classification. However, DNN's require a fixed vector input because the architecture of DNN's must be fixed and the number of input nodes cannot change unless a new DNN is created. This means DNN's cannot be used to solve this type of natural language processing problem because each input utterances contains a different number of words and therefore a fixed architecture is not ideal for this problem.

On top of this, DNN's do not distinguish or are not effected by the order of the input sequence. For example, assuming a DNN takes as input 6 words of fixed vector lengths, the DNN will not distinguish between the following two inputs even though they have different meanings.

- "The neural network made a man"
- "The man made a neural network"

The Recurrent Neural Network (RNN) [7] however does not suffer from these issues. They take an input sequence one token at a time and produce an output on each token. In our case each word is a token. Concretely, they work by having two inputs: an activation vector which is passed from the previous time step and the input token for the current time step. It will then produce the activation vector for the next time step and an output for the current input token. The RNN does this by iterating the following equations:

$$a^{<t>} = g(w_{aa}a^{<t-1>} + w_{ax}x^{<t>} + b_a) \tag{1}$$

$$y^{<t>} = g(w_{ya}a^{<t>} + b_y) \tag{2}$$

g is usually a squashing function such as the sigmoid function or tanh function. The matrices and biases $w_{aa}, w_{ax}, w_{ya}, b_a, b_y$ are learned through the back-propagation through time algorithm [17]. We can use models based on the RNN to solve the multi-dialogue conversation problem.

2.2 Sequence to Sequence Learning

This will be used as the base model. Sequence to Sequence learning uses Long Short Term Memory (LSTM) [4]. Although an output at a particular time step in an RNN is effected somewhat by the inputs in all the previous time steps via the activation vector, RNN's suffer from short term memory as they are not able to remember long term dependencies in sequences. This is due to the nature of RNN's and the vanishing gradient problem which occurs during back-propagation.

This is a big problem when sequencing a sentence because a word at the beginning of a sentence will often affect the choice of words later on towards the end of the sentence, this is what is meant by a long term dependency. LSTM solves this problem by learning long term dependencies in input sequences. The LSTM works by iterating the following equations:

$$\widetilde{c}^{<t>} = \tanh(w_c \left[a^{<t-1>}, x^{<t>}\right] + b_c) \tag{3}$$

$$i_t = \sigma(w_i \left[a^{<t-1>}, x^{<t>}\right] + b_i) \tag{4}$$

$$f_t = \sigma(w_f \left[a^{<t-1>}, x^{<t>}\right] + b_f) \tag{5}$$

$$o_t = \sigma(w_o \left[a^{<t-1>}, x^{<t>}\right] + b_o) \tag{6}$$

$$c^{<t>} = i_t * \widetilde{c}^{<t>} + f_t * c^{<t-1>} \tag{7}$$

$$a^{<t>} = o_t * \tanh(c^{<t>}) \tag{8}$$

Sequence to Sequence learning works by firstly having an encoder LSTM which reads the input sequence one token at each time step to produce a fixed length activation vector which is then passed to a decoder LSTM which will produce an output word at each time step by taking the prediction at the previous time step as an input until the end of sequence token $< EOS >$ is predicted. Using this model we are able to create a many to many RNN where the number of input tokens in a sequence is not the same as the number of output tokens and have no direct link.

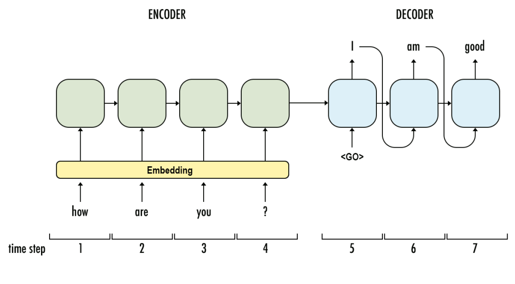


Figure 2: Example of a Sequence to Sequence model choosing a reply to an incoming message.

Within the decoder a softmax activation function over all words in the vocabulary is performed to choose the word with highest likelihood of being the next word in the output sequence.

$$y^{<t>} = \underset{softmax(w_{ya}a^{<t>})}{\arg\max} \tag{9}$$

2.3 The Speaker Model

As stated before, the Sequence to Sequence model suffers from a lack of consistent persona. The Speaker model is built on top of the Sequence to Sequence model to alleviate this problem. Firstly the speaker model creates vectors $v_i \,\epsilon\, \mathbb{R}^{K \times 1}$ where K is the dimensionality of word embeddings. Each v_i represents a single persona present in the data set. For example, in the Twitter Persona data set [6] responses are not labelled with persona descriptions (e.g. age, gender, occupation) as this would make the data set very large. Instead each unique twitter user in the data-set is given a vector v_i, $i \,\epsilon\, [1..N]$ which will represent their persona. v_i is learned by the model through back propagation alongside the other model parameters.

The vector v_i, corresponding to the given speaker, and is used along with the activation vector and the target representation generated in the previous timestamp to generate the current output representation within the decoder. The encoder remains unchanged. The speaker embedding v_i is shared accross all conversations that involve speaker i. The decoder LSTM will iterate over the following equations:

$$\widetilde{c}^{<t>} = \tanh(w_c. \left[a^{<t-1>}, x^{<t>}, v_i\right] + b_c) \tag{10}$$

$$i_t = \sigma(w_i. \left[a^{<t-1>}, x^{<t>}, v_i\right] + b_i) \tag{11}$$

$$f_t = \sigma(w_f. \left[a^{<t-1>}, x^{<t>}, v_i\right] + b_f) \tag{12}$$

$$o_t = \sigma(w_o. \left[a^{<t-1>}, x^{<t>}, v_i\right] + b_o) \tag{13}$$

$$c^{<t>} = i_t * \widetilde{c}^{<t>} + f_t * c^{<t-1>} \tag{14}$$

$$a^{<t>} = o_t * \tanh(c^{<t>}) \tag{15}$$

By doing this, the model learns the representation for each speaker; it can infer answers to certain questions about a given speaker even if the question has not been explicitly answered in the context of the given user using the answers for similar users.

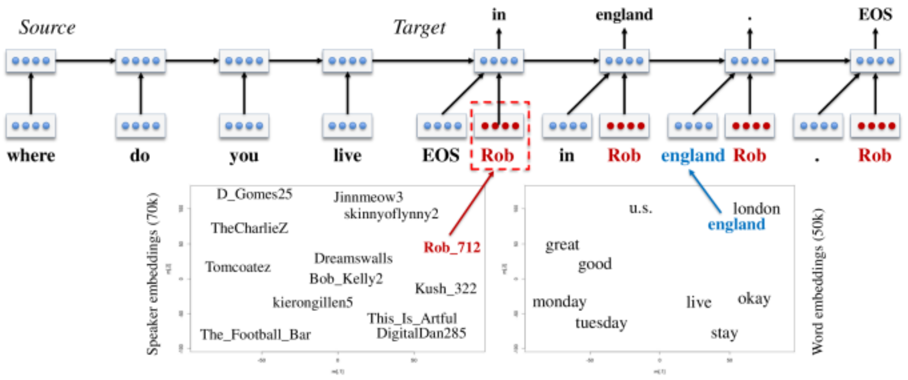


Figure 3: "Illustrative example of the Speaker Model introduced in this work. Speaker IDs close in embedding space tend to respond in the same manner. These speaker embeddings are learned jointly with word embeddings and all other parameters of the neural model via backpropagation. In this example, say Rob is a speaker clustered with people who often mention England in the training data, then the generation of the token 'england' at time $t = 2$ would be much more likely than that of 'u.s.'. A non-persona model would prefer generating in the u.s. if 'u.s.' is more represented in the training data across all speakers" [6].

For datasets which do have persona data annonated on top of each conversation (for example the PERSONA-CHAT data-set), the persona as a set of sentences can be prepended to the input sequence. [18]

2.4 Attention

Attention is one of the most influential and powerful ideas in deep learning to recent date. It was originally introduced in 2014 by Bahdanau et al [2] to improve neural machine translation however the concepts described in this paper can be applied to any Sequence to Sequence model. The problem with vanilla Sequence to Sequence is that the encoder is forced to encode the meaning or context of a input sentence into a fixed length vector which is passed to the decoder. Therefore it will struggle to deal with long sentences. Especially sentences longer than those found in the training set because the encoder will find it very challenging to include the whole context of the input sentence into a fixed length vector, some information in the input sentence will be lost.

Attention fixes this issue by allowing the decoder at each time step to choose a set of adjacent words within the input sequence to focus and put its 'attention' on. Therefore the decoder can get information about the input sequence not only from a fixed length vector produced by the encoder, but also the relevant context within the input sequence.

2.4.1 Encoder

Firstly lets look at how the Attention Model modifies the encoder in a standard Sequence to Sequence Model. A bi-directional recurrent neural network (BRNN) is used within the encoder. BRNN's work by forward propagating the input sequence through the network as usual but once the input sequence has been fully read, it will be read one token at a time in reverse to produce another set of hidden activation vectors which was obtained by reading the input sequence in reverse order one token at a time. Now each token within the input sequence is associated with two separate hidden activation vectors instead of only one. They can be concatenated together to make one fixed length vector. By doing this, at each time step the encoder is not only capturing information about previous words but also following words within the current context.

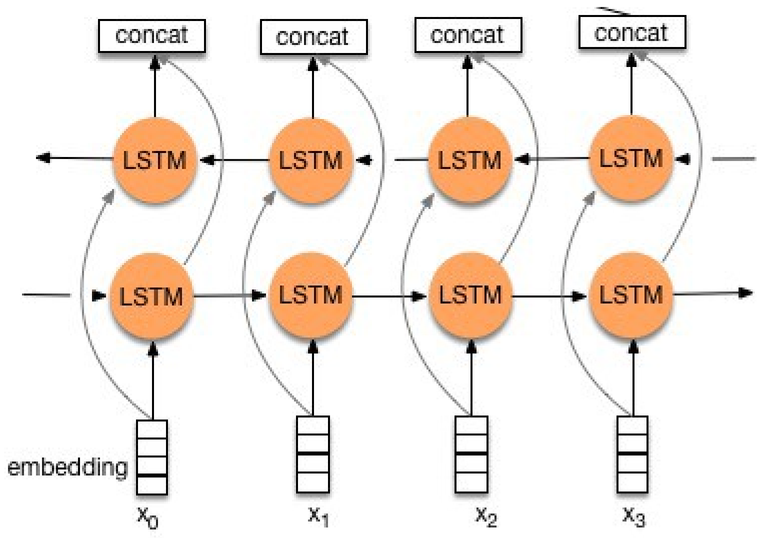


Figure 4: Illustration of the BRNN. It has a forward layer and backwards layer. The forward layer is computed through forward propagation of the input sequence. The reverse layer is computed through forward propagation of the reverse input sequence.

The concatenated hidden activation vector at each time step t within the encoder and the hidden activation vector for the decoder will be denoted as $h^{<t>}$ and $s^{<t>}$ respectively. The context vector at time step t within the decoder will be be denoted as $c^{<t>}$

2.4.2 Decoder

The decoder at time step t will will produce output y_t which will be the highest probability from:

$$p(y^{<t>}|y^{<1>}, ..., y^{<t-1>}, X) = g(y^{<t-1>}, s^{<t>}, c^t) \tag{16}$$

$$s^{<t>} = f(y^{<t-1>}, s^{<t-1>}, c^{<t>}) \tag{17}$$

Notice the addition of a context vector to the decoder. This context vector aims to capture the most relevant hidden activation vectors within the encoder. The context vector at each time step is calculated by performing a weighted sum over all hidden activation vectors obtained from the encoder.

$$c^{<t>} = \sum_{j=1}^{Tx} \alpha^{<t,j>} h^{<j>} \tag{18}$$

each weight $\alpha^{<t,j>}$ represents how relevant as a score between zero and one the input word j is to the current output at time step t. The weights over all hidden activation vectors in the encoder for the current decoder time step is computed by a softmax function.

$$\alpha^{<t,j>} = \frac{exp(e^{<t,j>})}{\sum_{i=1}^{Tx} exp(e^{<t,i>})} \tag{19}$$

$e^{<t,j>}$ describes the amount of attention $y^{<t>}$ should pay to encoder hidden activation vector $h^{<j>}$. This function of attention is not known before hand as it varies between data sets and tasks but it is learned by a small feed-forward artificial neural network usually with only one hidden layer. Parameters for this network are learnt jointly with the Attention Model by back propagating gradients of a cost function.

$$e^{<t,j>} = a(s^{<t-1>}, h^{<j>}) \tag{20}$$

The impact of Attention on BLUE scores for neuro machine translation can be seen in the figure below. BLUE is an automatic metric for determining the quality of an output translation given the input sequence [9].

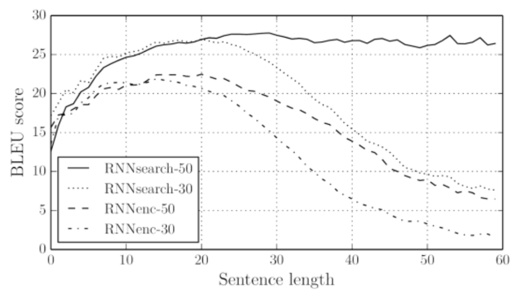


Figure 5: RNNsearch-50 was trained with Attention and maintains a consistent BLUE score

2.5 Deep Recurrent Neural Network

The Models described above can be extended by using a deep LSTM network [3]. A deep LSTM network works in a similar manner to a single layer LSTM network, the only difference is that LSTM's are stacked on top of each other within each time step. The output of an LSTM will be fed as the input to the LSTM in the next layer at each time step. Stacking LSTM hidden layers makes the model deeper and can be seen as recombining the learned representation from prior layers to create new representations at high levels of abstraction.

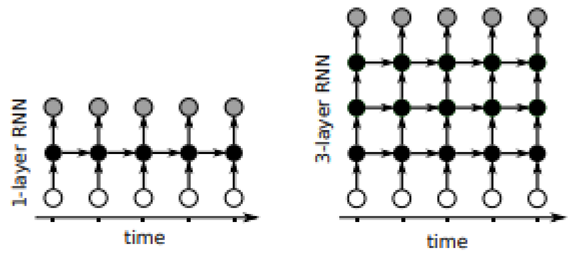


Figure 6: A visualisation of a single layer RNN and a three layer RNN [3].

2.6 Glove Embeddings

Word embeddings provide a method for represented words as fixed length vectors. The model will have a vocabulary of words and each word could be represented as a one hot vector. The one hot vector will have a dimensionality of K where K is the vocabularly size. For example, with a vocabularly size of 50,000, a word can be expressed as a one hot vector with dimensionality 50,000. $o \in \mathbb{R}^{50,000 \times 1}$

The issue with one hot vector representation is that words with similar meanings will usually have very different vector representations. What this means is that they will be far apart within the vector space. For example the words ”king” and ”queen” have index position 1,500, 30,000 respectively within a vocabulary list. The one hot vector representations will be markedly different from each other even though they have similar meanings. This will make it increasingly hard for any model to learn correct output sequences.

Word embeddings solve this issue by learning an embedding matrix $E \in \mathbb{R}^{N \times K}$ where N is the chosen length of each embedding (usually 512) and K is the vocabulary size. In this way each word in the vocabulary gets a unique word embedding with dimensionality N. The word embedding will group words with similar meanings so they are close together. This grouping can be visualised using the t-SNE algorithm [8].

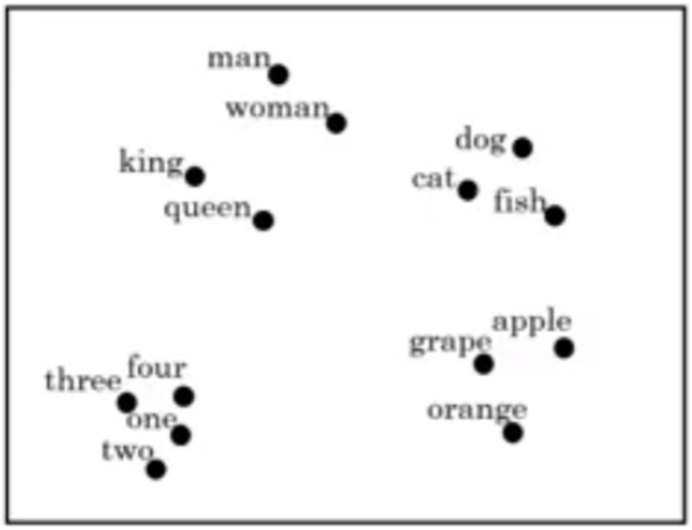


Figure 7: Dimensionality reduction using t-SNE to visualize proximity of similar words within the vector space.

There are various different models which can learn the embedding matrix E based on a large text corpus using unsupervised methods. Global vectors for word representation (Glove) [10] will be used to obtain the embedding matrix to be used in our model. This is because there is available a pre-trained embedding matrix learned by Glove on a text corpus of over 1 billion words.

2.7 Learning Framework

Implementing the model from first principles is unrealistic and unneeded, especially because there various different machine learning frameworks available to create, train and test models. For this project Tensorflow and Keras will be used on top of Python. Tensorflow is a flexible and highly optimised framework used by researchers and in production to create large scale machine learning systems capable of running on a variety of different devices. For example, mobile phones, PC's and GPU rigs. Because of it's wide use it also has great support and is a mature framework.

Keras is an interface built on top of Tensorflow, providing a higher level of abstraction to more easily build common components of a model. For example Keras provides an interface to create an LSTM easily. Whereas using Tensorflow alone would require the individual matrix operations to make up the LSTM to be specified instead. However Tensorflow is still required because it provides much more flexibility then Keras. In conclusion they can be used in conjuction with eachother to build the model.

2.8 Data Set

The data-set to be used will be the PERSONA-CHAT data-set [18]. This data-set includes a large corpus of conversations between two individuals. Each individual is given a persona as a set of 5 sentences describing background facts. The individuals are then given the task to get to know each other. This resulted in over 140,000 utterances, 19,000 dialogues and 1,355 different personas.

	# examples	# dialogues	# personas
Training Set	131,438	17,878	1,155
Validation Set	7,801	1,000	100
Test Set	6,634	1,015	100

Table 1: Number of utterances, dialogues and personas collected in the data set.

2.9 Evaluation

Both automatic metrics and human evaluation will be used to quantify the effectiveness of the chat-bot. The following automatic metrics will be used:

1. **Perplexity:** This is a measurement of how well the model predicts test data. Intuitively perplexity means the inability to deal with or understand something complicated or unaccountable, so the lower the perplexity the better. It is calculated by performing the log likelihood of the correct sequence.

2. **hits@1:** This meassures the next utterance classification loss. It works by choosing a random number of distractor responses from other dialogues. Then the model will select the best response among them, resulting in a score of one if the model chooses the correct response, and zero otherwise.

A human will converse with the chat-bot for a total of 4-6 utterances. After this the human will be asked to rate the chat-bot based on the following metrics:

1. **Fluency:** This measures between 0-5 how much the chat-bot answers make sense given the context of the conversation.

2. **Engagingness:** This measures between 0-5 how enganging the chat-bot is during the conversation.

3. **Consistency:** This measures between 0-5 how consistent the chat-bot is with its answers during the conversation.

4. **Persona Detection:** At the end of the conversation the human will be given a random persona within from the list of persona's and the persona which the chat-bot was using for the conversation. They will then have to predict which persona the chat-bot was using.

When obtaining the human evaluation metrics humans will not be aware if they are talking to a chat-bot or another real human. By doing this human evaluation metrics can be collected for a real life human. This is useful because we can compare these to the metrics of the chat-bot while removing any bias.
Overall it's difficult for a chat-bot to achieve good scores for all these metrics simultaneously, however the main goal of this project it to try to maximize persona detection and consistency as much as possible.

2.10 Voice Interface

Speech to text and vice versa are also machine learning problems that can be solved with sequence to sequence models however creating a separate model for this problem is outside the scope of this project.
Instead Google's speech recognition API can be used. The API is reliable and produces accurate results and so is a good match for this project. The user will not be forced to use the voice interface as some users may not have a microphone or simply prefer to type there questions when interacting with the social-bot.

3 Related work

Open dialogue conversation systems have a long history. ELIZA was the first program created for this purpose [16]. It worked by using a list of hard-coded rules to choose a response to a given message. This was not so effective because the chat-bot was very predictable in its answers.

Other approaches built on this by building statistical models and heuristics to work alongside the hard-coded rules. Another possible approach involves treating the generation of conversation dialogue as an statistical machine translation (SMT) problem [11]. It's only in recent years where an end-to-end, data-driven approach with neural language models have been adopted with great success.

4 Project Requirements

4.1 Functional Requirements

The following functional requirements specify what the project must do in order to be measured as successful.

1. **Language:** The project will use the Python programming language along-side machine learning frameworks Tensorflow and Keras.
2. **Seq2Seq:** The project will implement the Sequence to Sequence model which will provide a base model for conversation dialogue generation.
3. **Speaker Model:** The project will implement the Speaker Model in order to maintain a consistent persona.
4. **Attention:** The project will add Attention to the model so that the model can deal with long input sequences effectively.
5. **Website:** There will be a website to allow users from the internet to chat with the social-bot.
6. **User Interface:** The user interface will allow the user to enter a message to the social-bot in a text field in which case the chat-bot will send back a meaningful reply. The conversation history for the current dialogue will also be shown for the duration of the interaction.
7. **Voice Interface:** The user will also be given the option to speak to the chat-bot through a microphone. The audio will be converted to text and then processed by the model. When the reply is ready it will be converted to audio and played back to the user.

4.2 Non-functional Requirements

Non-functional requirements specify performance and control metrics which need to be met in order for the project to be measured as successful.

1. **Reply Speed:** The user should not have to wait more than 3 seconds for the chat-bot to generate a reply. A waiting time longer than this will make the conversation unnatural and cause users to become impatient.

2. **Website Traffic:** The website should be able to handle 10 concurrent users interacting with the chat-bot.

3. **Availability:** The website should have an up time of at least 99%.

4. **Maintainability:** The source code should be well documented, clear and concise. Documentation will take the form of well structured comments and PythonDoc.

5. **Reliability:** The website should not crash. If errors do occur they will be dealt with gracefully.

6. **Mobile Friendly:** The website will be built with a mobile first approach to ensure the website is compatible with a range of devices from mobile phones to tablets to large screen PC's.

5 Design

This sort of project is much more suited to being deployed on a website. This is because users can easily interact with the chat-bot whenever they want without the downside of having to install or download the software.
Furthermore, the chat-bot will be totally platform independent which means anyone using any device can access the chat-bot through the website. The only disadvantage to this is that users will need an internet connection to interact with the social-bot but this isn't a huge problem since the majority of devices are connected to the internet all the time anyway in this day and age.

The website will use Python for the back-end, specifically the Django framework to process incoming messages to the chat-bot. This is because the trained models created using Tensorflow and Keras can be saved and then loaded straight into the Django back-end. Other back-end languages such as PHP or C# were considered but using these languages would require an extra step of exposing an API to access the trained machine learning model which would add extra unnecessary work to the project. The front-end will use HTML, CSS and JavaScript as usual.

6 Implementation

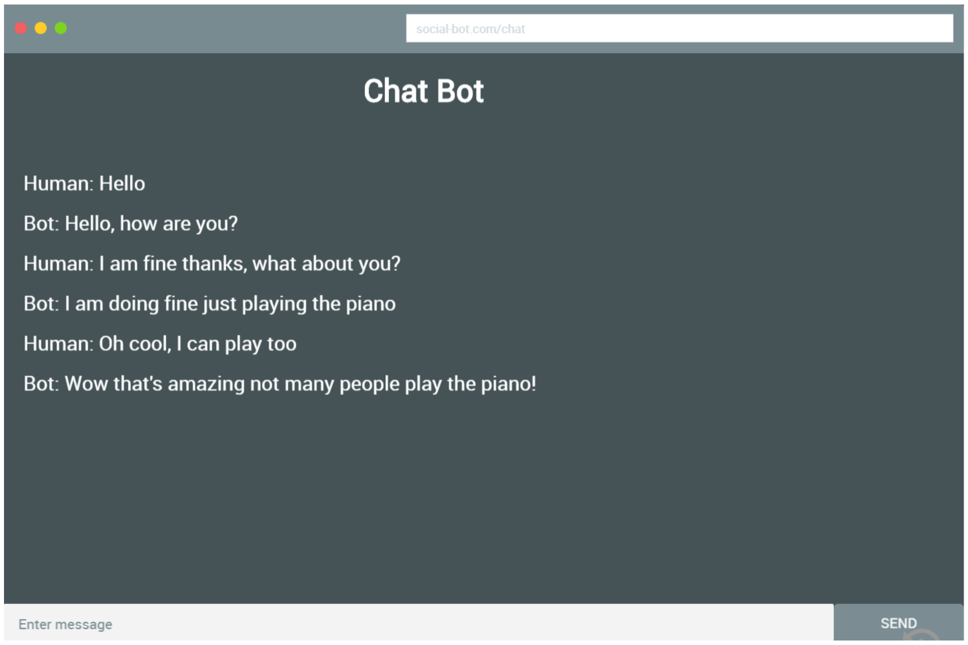
So far there is only partial implementations of the Sequence to Sequence model and some front-end user interface design. The majority of the work to date has been research specifically finding out which models are best suited to solving the problem at hand and how they can be applied.

6.1 Sequence to Sequence Model

There is a partial implementation of the sequence to sequence model. However most of the work is currently focused on pre-processing on the PERSONA-CHAT data set. The data set is given as a .txt extension, it's formatting is not clear or documented and therefore extracting conversations from the data set seems to be a non-trvial task.

6.2 GUI Design

As stated earlier the front-end will be on a single page website. The GUI should be simple and intuitive with a minimal number of feature. It should simply allow the user to enter a message either through typing or a microphone then the chat-bot will display its reply.



Human: Hello

Bot: Hello, how are you?

Human: I am fine thanks, what about you?

Bot: I am doing fine just playing the piano

Human: Oh cool, I can play too

Bot: Wow that's amazing not many people play the piano!

Figure 8: Simple wire-frame for the GUI.

7 Progress

In this section the current progress is reviewed critically against the original time plan within the project proposal. The project is on track with the original time plan and as predicted the first few months of the project were mainly spent researching and grasping machine learning concepts as there was no prior machine learning knowledge.

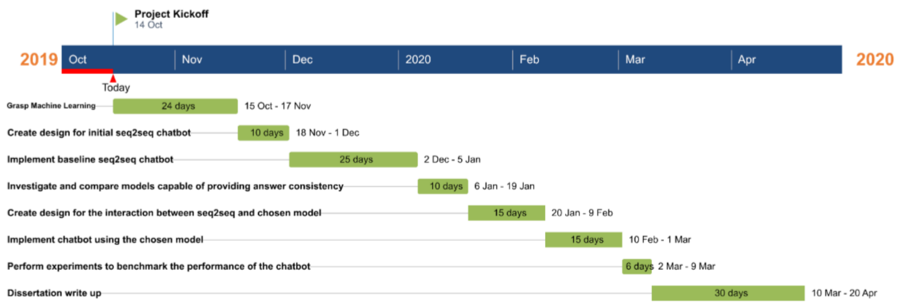


Figure 9: Gannt chart for the original time plan within the project proposal.

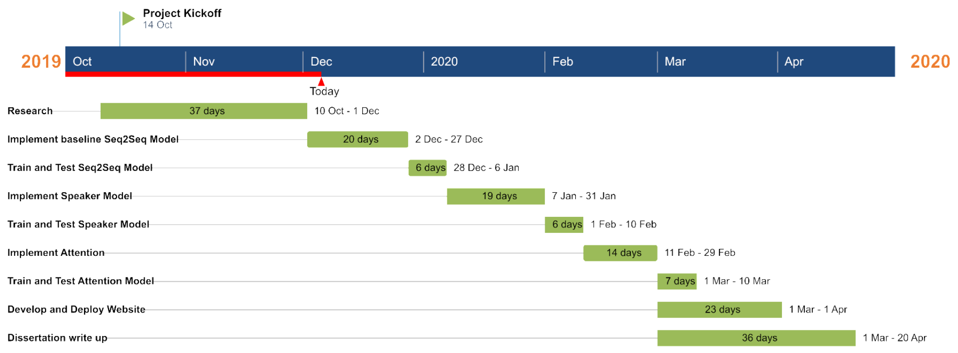


Figure 10: Gannt chart for the new time plan.

However it is clear that the original time plan is missing some development steps. Exactly which model to use on top of Sequence to Sequence also wasn't included because it was not known at the time since further researcher needed to be done. At the time it was assumed only one model would be built on top of sequence to sequence but as stated previously both The Speaker Model and Attention will be implemented.

Moreover, time taken to implement a model was considered but the time it takes to train a model was not. Previously it was assumed this wouldn't be very long but with the size of the data set being used, it could take up to one week to train a model. Additionally development of the website was not included in the previous time plan. Therefore the current time plan needs to be revised to include all the new tasks which need to be done. The new timeplan prioritises creating the models first before the website. This is because the trained models are the most important part of the project. Bugs and other factors which may slow down the project are also much more likely to occur during the development of the models whereas developing the website is less of a priority and for the most part straightforward. For this reason it is left till last.

In conclusion the project is on schedule. The workload will increase for the remainder of the project and this is reflected in the new project time plan. However, this is fine because of my module choice, the first semester is much heavier and so it's expected more work can be achieved within the second semester. It would be nice to be ahead of schedule because this would open up the possibility of exploring other models such as the Generative Transformer [14] which can achieve better results and achieve fast training times due to its parallelisation.

8 Bibliography

References

[1] Amazon. *Alexa Prize.* 2018. URL: `https://developer.amazon.com/alexaprize`.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2014).

[3] Michiel Hermans and Benjamin Schrauwen. "Training and Analysing Deep Recurrent Neural Networks". In: *Advances in Neural Information Processing Systems 26.* Ed. by C. J. C. Burges et al. Curran Associates, Inc., 2013, pp. 190–198. URL: `http://papers.nips.cc/paper/5166-training-and-analysing-deep-recurrent-neural-networks.pdf`.

[4] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-term Memory". In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: `10.1162/neco.1997.9.8.1735`.

[5] Jiwei Li et al. "A Diversity-Promoting Objective Function for Neural Conversation Models". In: *CoRR* abs/1510.03055 (2015). arXiv: `1510.03055`. URL: `http://arxiv.org/abs/1510.03055`.

[6] Jiwei Li et al. "A Persona-Based Neural Conversation Model". In: *CoRR* abs/1603.06155 (2016). arXiv: `1603.06155`. URL: `http://arxiv.org/abs/1603.06155`.

[7] Zachary Chase Lipton. "A Critical Review of Recurrent Neural Networks for Sequence Learning". In: *CoRR* abs/1506.00019 (2015). arXiv: `1506.00019`. URL: `http://arxiv.org/abs/1506.00019`.

[8] Laurens van der Maaten and Geoffrey Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605. URL: `http://www.jmlr.org/papers/v9/vandermaaten08a.html`.

[9] Kishore Papineni et al. “Bleu: a Method for Automatic Evaluation of Machine Translation”. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: `10.3115/1073083.1073135`. URL: `https://www.aclweb.org/anthology/P02-1040`.

[10] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “Glove: Global vectors for word representation”. In: *In EMNLP*. 2014.

[11] Alan Ritter, Colin Cherry, and William B. Dolan. “Data-Driven Response Generation in Social Media”. In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Edinburgh, Scotland, UK.: Association for Computational Linguistics, July 2011, pp. 583–593. URL: `https://www.aclweb.org/anthology/D11-1054`.

[12] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 0893-6080. DOI: `https://doi.org/10.1016/j.neunet.2014.09.003`. URL: `http://www.sciencedirect.com/science/article/pii/S0893608014002135`.

[13] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”. In: *CoRR* abs/1409.3215 (2014). arXiv: `1409.3215`. URL: `http://arxiv.org/abs/1409.3215`.

[14] Ashish Vaswani et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: `1706.03762`. URL: `http://arxiv.org/abs/1706.03762`.

[15] Oriol Vinyals and Quoc V. Le. “A Neural Conversational Model”. In: *CoRR* abs/1506.05869 (2015). arXiv: `1506.05869`. URL: `http://arxiv.org/abs/1506.05869`.

[16] Joseph Weizenbaum. “ELIZA - a computer program for the study of natural language communication between man and machine.” In: *Commun. ACM* 9.1 (1966), pp. 36–45. URL: `http://dblp.uni-trier.de/db/journals/cacm/cacm9.html#Weizenbaum66`.

[17] P. J. Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78.10 (Oct. 1990), pp. 1550–1560. ISSN: 1558-2256. DOI: `10.1109/5.58337`.

[18] Saizheng Zhang et al. “Personalizing Dialogue Agents: I have a dog, do you have pets too?” In: *CoRR* abs/1801.07243 (2018). arXiv: `1801.07243`. URL: `http://arxiv.org/abs/1801.07243`.