

1 | Introduction

The first chatbot (or conversational agent) was developed in the 1960's. It simulated a psychologist by following a set of rules to generate questions by using the user's original message [49]. Today, several personal assistants have been developed and are commercially available, such as Amazon's Alexa or Apple's Siri. These virtual assistants are able to perform certain tasks such as playing music, setting an alarm or answering (factual) questions by the user. However, dealing with the social and creative aspects of human language is still challenging. Most of the time, the assistants are unable to hold meaningful conversations with a person, and unable to maintain engagingness over long-term interactions. Part of this problem may be due to the agent's limited or non-existent memory, resulting in the agent being unable to understand context dependent messages by the user. Moreover, building a relationship with the user is challenging due to lack of user specific memory.

Even though conversational agents are important and highly applicable to our society, commercially the focus has been more on virtual assistants. Virtual assistants are task-oriented agents whereas conversational agents are non-task-oriented. The difference is that task-oriented agents are designed to be interacted with briefly to fulfill a task (e.g. book a train ticket) whereas non-task-oriented agents are designed for extended conversations. A few example applications for conversational agents are:

- Fitness trainer, motivating users to exercise regularly [3].
- Helping people overcome addiction.
- Emotional support for people with loneliness or depression.
- Diverse applications to assist the growth of special needs children.
- Educational agents.

Three existing chatbot examples are Insomnobot3000, Endurance ChatBot and Replika. Insomnobot3000 is an agent that people with insomnia can text with in the middle of the night when none of their conversational partners may be awake¹. Endurance ChatBot is an open-source work in progress, where the aim is to create a robot-companion for people who suffer from Alzheimer's disease². Replika "The AI companion who cares", is an agent that provides emotional support, and is always there for people who feel lonely or sad or just want someone to talk with³. For these types of agents to be able to help the target group, it is vital that the users stay engaged long enough for the agents to help them, which illustrates the importance of research in the area of long-term engagingness for conversational agents. For the case of Endurance ChatBot, it is especially important that the chatbot has memory to be able to evaluate the state of the user's health status.

¹<http://insomnobot3000.com/>

²<http://endurancerobots.com/azbnmaterial/a-robot-companion-for-senior-people-and-patients-with-alzheimer-s-disease/>

³<https://replika.ai/>

The chatbots of today generate messages to users by either following a set of rules to return a template based on the recognized sentence pattern (rule-based), or by being trained on a large amount of (human-to-human) conversational data (corpus-based). The rule-based approach can be considered as the standard way of creating conversational agents in the past. This is because the previous winner in the Loebner Prize was a rule-based chatbot “Mitsuku”, for the 4th year in a row⁴. The Loebner Prize is a contest where developers test their bots against the Turing Test, and the one with the most human-like bot wins. For rule-based chatbots, this requires the creation of hundreds of templates to handle a broad range of inputs, which in turn costs time and money. Another popular method is to use a retrieval database, where a database of replies are kept and retrieved. The retrieval method works well when the user’s input is similar enough to entries in the database, as it may retrieve related, high quality, and potentially engaging replies. Once the user’s inputs are not similar to any entry in the database, then the flaws of the method are observed. The agent will then reply with something seemingly random, change topic, or give a safe reply like “I don’t know”. Both methods suffer from repetitiveness as there is only a limited number of available replies at any given time. The third method is to train neural models to generate response. Training a model with the help of machine learning is an attractive solution as it is cheaper than creating hundreds or thousands of templates, and compared to the retrieval structure, it is able to generate an answer to any user input. However, generative models instead suffer from inconsistency as well as having a tendency to produce non-specific answers such as “I don’t know”.

Inconsistency example,

where PERSON 1 is a human and PERSON 2 is a chatbot [56].

[PERSON 2:] I love to cook and bake but I rarely make time for it

[PERSON 1:] what do you usually bake?

[PERSON 2:] I do not cook, I bake cakes at a local restaurant. how about you?

The mentioned methods have their respective pros and cons, and therefore, this thesis aims to develop a chatbot with rule-based methods and corpus-based methods combined. With rules and templates, a memory structure will be introduced and used consistently. The corpus-based approach will support the templates to handle any input that is out of scope for the templates. Memory will be used for two main tasks in this thesis:

1. Memory based question generation to ask the user personal questions.
2. Personalization of the agent by storing personal background information about the agent.

By storing memory about the user, the agent will be able to, at a later time re-use information provided by the user in past interactions, to generate personal questions. By storing memory about the agent, the user will be able to ask questions and get to learn about the agent. Both of these tasks are researched for the purpose of contributing towards the field and the future development of personal social chatbots which are capable of long-term interaction; chatbots that users may develop a personal bond with.

1.1 Motivation

The social chatbots of today usually have short-term memory (memory for the current interaction) but have limited (if any) long-term memory (memory over several interactions). Without short-term memory it becomes difficult for the agents to understand context dependent user utterances, such as follow-up questions or utterances referring to “he/she/it”.

⁴<https://aisb.org.uk/mitsuku-wins-2019-loebner-prize-and-best-overall-chatbot-at-aisb-x/>

Without long-term memory it becomes difficult for the agent to build rapport with users. To build rapport it is necessary to have a personal background, remember information about the user, and then to find common ground such as shared interests.

Example of context dependent utterances:

User: What is your favorite sport?

Agent: I like football the most.

User: Why do you like *it*?

The project was carried out at the Human Media Interaction (HMI) research group at the University of Twente. HMI partly performs research in the area of dialogue and natural language understanding and generation. The research group is specifically interested in research aimed at social and creative applications, such as conversational agents or social chatbots.

1.2 Purpose

The purpose of this thesis is to investigate how long-term and short-term memory can be used in a chatbot to simulate a consistent persona for question answering and to enable long-term question asking via user modeling. This is done by implementing and testing a chatbot with user specific and agent specific memories, where long-term memory data is mainly used through rule-based methods, such as templates, and short-term memory is used in a generative model.

1.3 Research Questions

The main research question is to find out, *how can a personal social chatbot be developed with long and short-term memory, such that the interaction with the chatbot remains engaging over a longer period of time?* In line with the main research question, the current work aims to address the following sub-questions:

Development

1. How can the user and agent specific memory models be developed to extract and store information from user utterances, and apply them to agent utterances?
2. How can short-term memory be developed such that the agent can understand and provide a relevant answer to user utterances, such as follow-up questions, from the context of the conversation history?

User testing

1. How is the user's experience with an agent affected when the agent generates questions which are personalized based on the user's utterances from previous interactions?
2. How is the user's perception of the agent affected by the agent having a personal background, which allows for self-disclosure responses?
3. To what extent would the user's perception of the agent be affected if the user could ask the agent self-disclosure questions?

1.4 Delimitations

The conversational agent will be developed as a non-task-oriented system. That is, it will not be a type of question-answer (QA) bot, for example by searching online for factual answers in order to reply to the user, as Siri does. It is not aimed for it to perform tasks such as scheduling, setting reminders or alarm. The chatbot will be developed primarily to investigate what effects the memory has on users for question asking and question answering. As such, the final chatbot is not intended to be a stand-alone product that can be deployed outside of the scope of the user tests in the thesis. The chatbot will consist of two components, a question asking component and a question answering component.

The chatbot will focus solely on text input and output. Its main platform is targeted at communication applications, such as Messenger, Discord and WhatsApp, and websites in general, where text exchanges take place. Therefore, features related to speech will not be taken into consideration. However, this can be left as a part of potential future work.

It is expected that the user will send messages such that each message is limited to one intent. It is also expected that the users should use proper and grammatically correct English, although minor typographical errors are allowed.

The user and agent memory structure will be restricted to storing information on a limited range of general topics, such as sports, hobbies, movies, etc. The structure aims to retain memory about like/dislike preferences, e.g. that the agent likes apples but dislikes oranges, that the user's favorite animal is cat.

1.5 Thesis Structure

The rest of the report is structured as follows. Chapter 2 discusses the background of conversational agents, various models, a framework and a few useful resources. Chapter 3 describes the development of a question answering chatbot. The second phase of development of the question answering chatbot after a user test is presented in Chapter 4. Chapter 5 describes the development of a question asking chatbot. Chapter 6 explains development of an integrated chatbot, where the question answering chatbot and the question asking chatbot are combined. After that, Chapter 7 presents the results and compares performance across the three chatbots. Discussion on the results and methodology, and future works are presented in Chapter 8. Concluding remarks follow in Chapter 9.

2 | Background and Related Work

This chapter provides theory for key concepts relevant to this thesis, such as different types of chatbots; task-oriented and non-task-oriented, and the different methods to create chatbots, such as corpus-based and rule-based approaches. Related work is investigated to find which methods have been tested, and which methods contribute toward the development of engaging social chatbots. Additionally, the chapter provides an overview of existing methods for evaluating the performance of chatbots.

2.1 Conversational Agents

Chatbots can be classified into two types, based on their different natures of the interaction. One chatbot type is designed to help accomplish tasks such as answering frequently asked questions or making a reservation at a restaurant. This type of chatbot is referred as task-oriented dialogue agent [20]. The other chatbot type is for entertainment, conversation, building a relationship and such, hence usually carry on longer conversations with the users. These are usually categorized as non-task-oriented chatbots and is what the project will primarily be focusing on. Chatbots are developed through rule-based or corpus-based methods, sometimes in combination.

2.1.1 Rule-Based Methods

Rule-based chatbots, such as Mitsuku (2.4.2), make use of rules to generate responses, e.g. if given input a, then do action b and return response c. This consists of creating pattern-response pairs or templates. A pattern may be as follows "My name is *", where the * is a wildcard that can be re-used in the response, "Nice to meet you *". Templates can be created to handle a broader range of inputs by making rules around the semantic meaning with the help Natural Language Processing methods such as Semantic Role Labeling, Part of Speech tagging and Named Entity Recognition, but may require that the user inputs complete sentences [27]. Rule-based methods may be more time demanding than corpus-based methods as it requires the creation of many hand-written rules, but it may also be able to handle a wider range of topics because of it.

2.1.2 Corpus-Based Methods

Corpus-based (or data-driven) approaches make use of (human) written text data, usually in a human-to-human conversation setting, but also human-to-machine or simply by mining text online e.g. news articles [59]. The data can be used either for information-retrieval (IR) or for Machine Learning. In information retrieval systems, the data is maintained in a database, for example by storing key-value pairs such as question-answer pairs. A user's input becomes a query to the database where it retrieves the most relevant pair based on query and key similarity, and then the corresponding value is returned as a response. Information retrieval systems contain high quality responses for when there is a match between the user's

query and existing information in the database. The responses, however, are limited to the range of data and topics that exist in the database.

Different Machine Learning models have been developed for the task of text generation by training on large amounts of text data (generative models). The trained models can then generate an output to any input by the user, although the quality of the response is usually lower compared to IR systems. It is an attractive solution as it allows for the creation of chatbots simply by training a model on a dataset.

2.2 Generative Models

Generating outputs, such as text, is a sequential task. Deep Neural Networks (DNN) can learn vector representations of text and generate text (sequence of words) from representations. The fundamental neural network architecture which uses sequential information is the Recurrent Neural Network (RNN) architecture, which has also been previously used for text generation [43]. The network consists of a number of sequential hidden state vectors for each time-step (memory), making the network deep in the time direction. The hidden state at a specific time-step is calculated from the previous hidden state and the input at that time-step, which leads to every future hidden state having some information from the previous hidden state. The model is also cheap to train by a method called 'backpropagation through time' (BPTT), as it shares trainable weights over the entire sequence. A big issue with RNNs however, is that of information getting lost over time, due to the vanishing (or exploding) gradient problem¹. Simply put, gradients increase or decrease exponentially with respect to N , in an N -layer network, which leads to long training times and difficulties for the model to remember information from previous time-steps.

Two models that solve this problem are the Long Short-Term Memory (LSTM) model [16] and the Gated Recurrent Unit (GRU) model [8], which are designed to remember information over long periods of time by the use of "memory" units. GRUs are considered a slight variation of LSTMs but functions similarly. The main difference is that GRUs are slightly faster, due to a simpler architecture, for better and for worse [9]. LSTMs and GRUs are in turn used in recent generation-related architectures such as the Sequence-to-Sequence (Seq2Seq) model [44], which in turn may have inspired the Transformer [45] architecture which is used today (e.g. GPT-2).

2.2.1 Sentence and Word Embeddings

Word embeddings, e.g. Word2vec [28], GloVe [31], etc. are vector representations of words found through neural networks. Another description is that word embeddings allow for representing a word as a vector of real numbers. The embeddings are the result of training a neural network to predict the probability of a word in a context (continuous bag-of-words). A word embedding can express semantic and syntactic aspects of the word to some extent. Word vectors also exhibit additive composition [28], e.g. "king - man + woman \approx queen". One application for word embeddings is in neural networks by encoding the words as vectors to be used as input (and potentially as output).

Cosine similarity is a metric commonly used in the context of Natural Language Processing to measure similarity of texts. The similarity is found by transforming the texts into normalized vectors and thereafter calculating the cosine of the angle between the two vectors (or texts). As such, by using word embeddings it is possible to calculate how similar one word is to another, by using cosine similarity. This is possible because the models (e.g. Word2vec) learn which words commonly occur in the same context, which then leads to the words being closer together in vector space. "You shall know the word by the company it

¹http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf

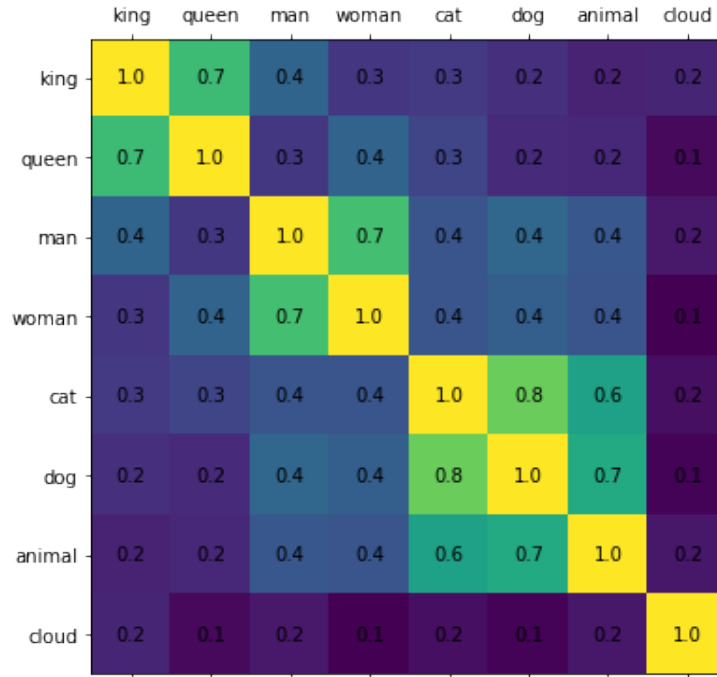


Figure 2.1: Word embedding similarity matrix using spaCy's tokenizer and similarity function.

keeps". The closer the words are in the vector space, the higher the cosine similarity. The range of similarity is between 0.0 and 1.0 or sometimes from -1.0 to 1.0, where a value close to 1.0 means that the texts are identical or very similar. Given the word embeddings of the words king, queen, cat and dog, the embedding of king will be closer to the embedding of queen than it will be to cat or dog, see figure 2.1. Similarly, the embedding of cat will be closer to the embedding of dog.

Extending the concept of word embeddings for words into sentence embeddings for sentences, it is possible to calculate the cosine similarity of two sentences. An example of a model that produces sentence embeddings is the Universal Sentence encoder [6] developed at Google. Finding the similarity between sentences may be useful as part of clustering (3.4), answer ranking (4.4.4) or similar applications. As such, word and sentence embeddings as well as cosine similarity will be utilized throughout this thesis.

2.2.2 Fine-Tuning and Transfer Learning

The terms 'fine-tuning' and 'transfer learning' have been observed to be used interchangeably. The concept is to use an existing model that has been pre-trained on a specific task by using a large dataset, and then applying it to a different but related task by fine-tuning the model or using transfer learning [29]. The difference is that fine-tuning uses the trained weights of the pre-trained model as a starting point and continues training the weights on a new dataset. Transfer learning has more technical implications, as the pre-trained model is used in some way as part of a new model. One example of transfer learning would be to train a new model on a different dataset and use the pre-trained model in a pipeline with the new model, see figure 2.2.

The concept is useful when there is a significantly lower amount of data available for a sub-task than to a more general task. Consider the general task of recognizing vehicles in images and the sub-task of classifying what type of vehicle it is (e.g. truck or car). The

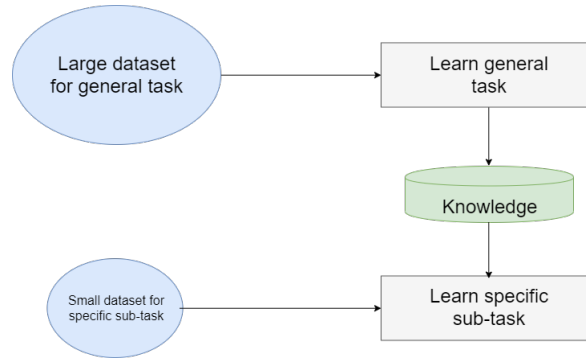


Figure 2.2: Transfer learning example where the general task model’s knowledge can be transferred to the sub-task model

labeled data (assuming supervised learning) for recognizing vehicles may be significantly larger than a dataset that specifies the vehicle type. As such, the smaller dataset, if used to train a model from scratch may not be able to train the model to recognize vehicles as well as the larger dataset. Therefore, by first training a model on the larger dataset for the general task, and then using the pre-trained model to fine-tune for the specific sub-task, the results may be better. The same concept can be applied for the task of text generation. First, a model is pre-trained to understand and generate coherent text by using a large dataset with many different types of human-written text, and then fine-tuned for the sub-task of dialogue [51].

The concept of fine-tuning will be utilized in this thesis on existing large pre-trained generative models trained for text generation, by fine-tuning the models on the task of question answering.

2.2.3 Seq2Seq or Encoder-Decoder

The Sequence-to-Sequence (Seq2Seq), also known as Encoder-Decoder model, was first introduced by researchers at Google in 2014. Seq2Seq is a method that makes use of the Long Short-Term Memory (LSTM) model in an encoder and decoder architecture [44]. It takes as input a sequence of words and generates an output sequence of words. The encoder reads the input sequence one time-step at a time and maps the sequence to a vector of large fixed-dimensional vector representation (e.g. word embedding (2.2.1)). The decoder then gets information from the encoder to decode the next output, while considering the decoder’s previous output. The model obtained a BLEU score of 34.81 on the WMT’14 English to French translation task, which was the best result achieved by direct translation with large neural networks at the time of its release (2014). BLEU is common metric used for evaluation in machine translation, based on word-overlap [30].

The release of the model led to researchers training and testing the model as a conversational agent [46]. The researchers were able to train an end-to-end conversational agent by using a data-driven approach. End-to-end training in this case means the ability to train the model into a conversational agent from scratch by using a dataset. Although it was possible to train the model for the purpose of acting as a chatbot, the model tended to give short and unengaging responses. Additionally, the model did not capture a consistent personality. The result can be illustrated by one-turn interactions, extracted from a longer conversation. The flaws of short and dull answers, and inconsistent personality, have since been further researched in [23] and [24], respectively.

Human-Agent one turn interaction

Human: what do you like to talk about ?

Machine: nothing .

Human-Agent inconsistency

Human: what is your job ?

Machine: i 'm a lawyer .

Human: what do you do ?

Machine: i 'm a doctor .

Today, several tutorials exist that illustrate how a chatbot can easily be created by using Seq2Seq². A variation of the model was also used by Microsoft as part of their chatbot, XiaoIce [59] (2.4.4). While the Seq2Seq model have been used in recent years in chatbot applications, it will not be used in this thesis due to the model being superseded by Transformer models. At the time of writing, Transformer type models have state-of-the-art results on several Natural Language Processing tasks.

2.2.4 Transformer

The Transformer model was first introduced in the paper "Attention Is All You Need" [45] in 2017, where it hit new records on the WMT'14 translation task. The model made use of the encoder-decoder concept as seen in Seq2Seq but discarded the concept of using RNN by instead using an attention mechanism and feed forward neural networks. The attention mechanism allows the decoder to have access to all hidden vectors in the input, which allows for better context understanding. At each time-step in the decoder, a prediction of the next word output is made by using the weighted sum of all hidden vectors in the input. Additionally, the model's architecture allowed for parallelization which led to faster training times. Since it no longer uses RNN, and computation is parallelized, positional encoding is used to inform the model of the word order of sentence sequences.

Since it was first introduced, the architecture has been used in several new models, most notoriously, Bidirectional Encoder Representations from Transformers (BERT) [11], and Generative Pretrained Transformer 2 (GPT-2). While the original transformer used an encoder-decoder structure, the BERT model only consisted of encoders³, while GPT-2 only consisted of decoders⁴. The architecture has led to several state-of-the-art models in the field of natural language processing, some of which score higher than their respective human baseline. The NLP benchmarks GLUE⁵, SuperGLUE⁶, SQuAD⁷, CoQA⁸ and QuAC⁹ are, at the time of writing, dominated by transformer models, most commonly with variations of BERT such as ALBERT and RoBERTa.

Transformer models can be abstracted into three groups, Sequence-to-Sequence, auto-regressive, and auto-encoding models. Sequence-to-Sequence models, such as the original Transformer, consist of a encoder and decoder part. The natural application is translation. Auto-regressive models, such as GPT-2, are pre-trained to predict the next token (word), given the previous inputs. They correspond to the decoder part of the original Transformer and the natural application is text generation. Auto-encoding models, such as BERT, are pre-trained by first masking input tokens to then attempt to reconstruct the original input. They correspond to the encoder part of the original Transformer and the natural application is sentence classification or token classification. All three groups of models can be trained for various NLP tasks depending on their characterization.

A library by Hugging Face Transformers contains several pre-trained state-of-the-art Transformer models¹⁰, such as GPT-2, BERT [11], Text-To-Text Transfer Transformer (T5) [33],

²https://pytorch.org/tutorials/beginner/chatbot_tutorial.html

³<http://jalamar.github.io/illustrated-bert/>

⁴<http://jalamar.github.io/illustrated-gpt2/>

⁵<https://gluebenchmark.com/leaderboard>

⁶<https://super.gluebenchmark.com/leaderboard>

⁷<https://rajpurkar.github.io/SQuAD-explorer/>

⁸<https://stanfordnlp.github.io/coqa/>

⁹<https://quac.ai/>

¹⁰https://huggingface.co/transformers/model_summary.html

etc. As such, the library enables comparison of different models. Hugging Face also hosts an online interactive demo of generative transformers¹¹.

This thesis will consider auto-regressive models (GPT-2) for text generation due to the availability of models which have been pre-trained with the text generative task in mind.

2.2.4.1 GPT-2

Generative Pretrained Transformer 2 (GPT-2) was released as a pre-trained transformer model in 2019 by OpenAI, following the previous model under the same name (GPT). The model achieved state of the art results on 7 language modeling datasets [32]. The main difference from the previous model is that the new model comes in different sizes and is trained on a larger dataset. The models released by OpenAI were pre-trained on 40GB of internet texts. [32]. The different versions are ‘124M’, ‘355M’, ‘774M’ and ‘1558M’ [39]. The names represent the size of the models, ‘124M’ consist of 124 million parameters whereas ‘1558M’ consists of 1.5 billion parameters. The models generally generate “better” texts as the number of parameters increases, as observed in a study by [39] on human detection of generated news articles. The models were given a credibility score in the range 1-10, where the score is composed of independent clarity, accuracy and believability scores. The 355M model had a mean credibility score of 6.07 while the 774M model scored 6.72 and the 1558M model scored 6.91. Although the score increases with the size of the models, the improvement from the 355M model to the 774M model is more significant than the improvement from the 774M model to the 1558M model in terms of the credibility score. The larger the model is, the longer it takes to fine-tune (2.2.2), and the longer it takes to generate text once the model is fine-tuned (A.3).

Following are three short, cherry-picked examples of text generated by the largest GPT-2 model, given an input prompt. One longer example is provided in the appendix A.1.

Text generation example using the 1558M model

Prompt: *What are your hobbies?*

Generated: I enjoy hiking, biking, and hanging out with my family. I also love to cook. I love to learn new skills, and I love to share my love of cooking with others.

Another example, with the input prompt in italic

Get rich quick with these 5 steps to get rich quick:

1. Write a check for \$100,000.
2. Donate the money to charity.
3. Get a job.
4. Get a divorce.
5. Be a millionaire.

Third text generation example

The steps to getting famous are: 1. Get a lot of people to know you 2. Start a website 3. Start a blog 4. Start a YouTube channel 5. Make a living from it 6. Be really good at something 7. Use your fame to get things done 8. Do stuff that will get you noticed 9. Make money from it 10. Get famous.

That’s a lot of steps to get famous, and I don’t think it’s going to happen to me.

2.2.4.2 Distillation

As language models, especially transformer models were becoming larger and larger, the concept of distillation was applied to create distilled versions of the large models. The method is to down-scale the architecture into a new (student) model, where this model is trained to reproduce the behavior of a larger (teacher) model. This resulted in smaller and faster versions

¹¹<https://transformer.huggingface.co/>

which still retain most of the larger model's capabilities. DistilBERT [37] is an example of distillation where the BERT model's size was reduced by 40% (from 110 million parameters to 66 million) and runs 60% faster, while retaining 97% of its language understanding capabilities. The concept was also applied to the RoBERTa model and the 124M sized GPT-2 model to create DistilRoBERTa and DistilGPT-2¹². Therefore, distilled models, such as DistilGPT-2, can be an alternative to considered to achieve faster text generation, if the larger models take too long to generate text in a chatbot setting.

2.2.4.3 Other Auto-Regressive Models

Other than the GPT models, XLNet, Transformer-xl, CTRL, are three additional models which are available through Hugging Face's transformer library. CTRL consists of 1.6B parameters and takes too long to generate text to be considered as a generative model for the thesis. XLNet improves upon the TransformerXL model and is a lot faster. Extensive testing was not performed with either XLNet or TransformerXL, although it seemed as if XLNet was significantly slower than GPT-2 at text generation, after using Hugging Face's API¹³. As such, a decision was made to use GPT-2 for text generation.

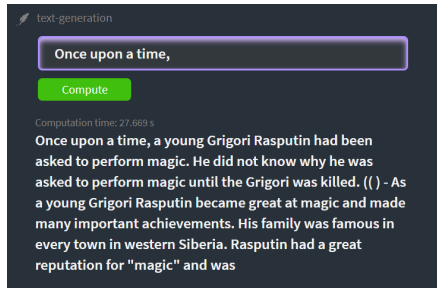


Figure 2.3: XLNet generating 59 words in 27.669 seconds.

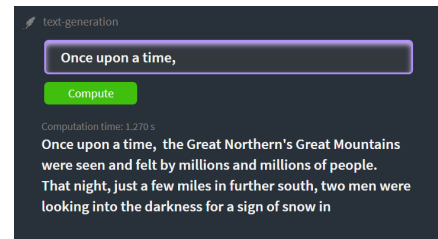


Figure 2.4: GPT-2 (124M) generating 37 words in 1.270 seconds

2.3 Persona-Chat Task and Agents

A dataset called Persona-Chat was developed with the aim to allow for the development of more engaging chatbots with consistent and identifiable personas [56]. The dataset consists of conversations between crowdworkers who were randomly paired and asked to act out a persona, based on a short description, while keeping the chat natural. The dataset was used in The Second Conversational Intelligence Challenge (ConvAI2)¹⁴ which was a competition held to find approaches to developing engaging chatbots capable of open-domain conversations [12]. The dataset and the competition aim towards finding the solutions to some of the common problems with chatbots. The considered problems are:

1. The lack of a consistent personality, due to the models being trained on data consisting of dialogues from different speakers [46] [24].
2. Lack of explicit long-term memory, due to being trained to produce utterances given only recent conversation history.
3. A tendency to produce vague, non-specific answers such as "I don't know", which are not engaging for human users [23].

¹²<https://github.com/huggingface/transformers/tree/master/examples/distillation>

¹³<https://huggingface.co/xlnet-base-cased>

¹⁴<http://convai.io/2018/>

The competition contained automatic and human evaluations. Human evaluation was through crowdworking, following a similar method as in the creation of the persona-chat dataset. The human users interacted with the agent for 4-6 dialogue turns, following a random persona provided to them. The users were then asked to answer the question "How much did you enjoy talking to this user?" on a scale of 1-4. The users were also tested whether they could distinguish the persona the model was using from a random one. Automatic evaluation was through three metrics:

1. Perplexity - a metric of text fluency.
2. F1 score - applied as a metric for word-overlap between generated response and gold response.
3. Accuracy of the next utterance selection when given random distractor responses mixed with the gold response (accuracy of ranking).

Hugging Face had the best performing agent (2.4.3) on the automatic evaluation and came second in the human evaluation, with an engagingness score of 2.67. The winner "Lost In Conversation" achieved a score of 3.11. Analysis was conducted to find how the highest scoring model from the automatic evaluation was bested in the human evaluation. The conclusion was that Hugging Face's model tended to ask too many questions, which disrupted the balance of question-asking and question-answering.

Automatic evaluations are still flawed in comparison to human evaluation. Notably, always replying with the response "i am you to do and your is like", would outperform the word-overlap measure of all models [12]. This is related to the findings that word-overlap metrics do not correlate well with human judgement [25]. When adding the last utterance of the conversation as one of the distractor responses for the utterance selection metric, it was observed that most models suffered, resulting in parroting. This illustrates that these models may have relied too much on candidate response ranking influenced by word-overlap to the previous utterance.

Through human evaluation of persona detection, Hugging Face achieved a score of 0.98 and Lost In Conversation achieved a score of 0.9. The human-to-human detection score of persona was at 0.96. It is clear that the persona can be recognized confidently. However, as models tend to repeat (persona) sentences, this may lead to a high persona detection score but lead to a lower engagingness score. As such, training models to use the persona to create engaging responses rather than copying/repeating it remains an open problem [12].

Final conclusions from the competition is that the best observed models (in the competition) were variations of the generative Transformer architecture. That being said, the competition was held before the BERT and GPT-2 model were released which may have further potential. Generative models worked better than retrieval models on this task (out of the observed models). Comparing retrieval and generative models without human judgement remain an open problem for the dialogue task. Models that do not have a balance of question-asking and question-answering in conversations may suffer from human evaluation in terms of engagingness.

Models still suffer from inconsistency such as generating responses like "I work as a snow-board instructor" followed by "I work for a food company". Additionally, models tend to ask questions that have already been answered previously. These may be solved through the development of a memory structure and/or with the help of Natural Language Inference [50]. The competition and the competing models are useful for this thesis both as a guideline as to how chit-chat models are developed, and for comparison of engagingness, allowing results of user tests to be compared to previous chit-chat agents.

2.4 Relevant Conversational Agents

Well-known or established social chatbots, such as Mitsuku, XiaoIce and Replika are investigated to find the pros and potential cons of the methods used in each chatbot.

2.4.1 Long-Term Engagingness

Two studies were conducted measuring the effects of agent behavior variability and the effects of back stories on subjects in the area of long-term interventions with relational agents [3]. The system was designed to motivate users to exercise, e.g. to take walks regularly. Their results were measured both by how often the participants interacted with the system and also by requesting the subjects to complete a questionnaire at the end of each interaction. The questionnaire for the first study measured the subjects desire to continue using the system "How much would you like to continue working with Karen?", and the repetitiveness "How repetitive are your conversations with Karen?". The second study had a questionnaire with three items asking how much they "enjoy the stories that the counselor tells", "look forward to talking to the counselor", and "feel that the counselor is dishonest". Each item was assessed on a 5-point rating scale ranging from "not at all" to "very much".

It was observed that techniques that made the agent more lifelike and human increased user engagement. Their first study showed that increased variability in their agent's behavior led to increased engagement. Their second study showed that personalizing the agent by giving it an artificial auto-biography also led to increased engagement. These findings motivate the chatbots in this thesis to be developed by considering aspects that may increase perceived human-likeness, such as memory, persona and variability.

2.4.2 Mitsuku

Mitsuku is a rule-based chatbot that is built by using Artificial Intelligence Markup Language (AIML). It is built on top of another famous chatbot called ALICE [48]. The creator of the chatbot, Steve Worswick, has worked on the agent a few hours at a time since around 2004¹⁵. Mitsuku has won the greatest number of Loebner prizes (5 times), where it was deemed as the most human-like chatbot in the competition.

Mitsuku has a limited memory where it saves information about the user and some contextual keywords, see figure 2.5. By using the keywords, the agent is able to understand a question like "What do you think about him?", after previously talking about Eminem. Some observed flaws with the chatbot are its repetitiveness, where it uses the same template answer repeatedly, and that it sometimes fails to extract information that it uses in future templates¹⁶. This thesis will partially utilize memory slots to store information about the user, similar to Mitsuku.

Human: I like eminem

Mitsuku: What is your favourite Eminem song? You like Eminem.

Human: I like several of his songs.

Mitsuku: I like The Bible. You like several of.

2.4.3 Hugging Face's Persona Chatbot

The company called Hugging Face participated in the ConvAI2 competition, where the aim of the competition was to find approaches to developing engaging chatbots capable of open-domain conversations [12]. A variation of the chatbot that was developed for the competi-

¹⁵http://aidreams.co.uk/forum/index.php?page=Steve_Worswick_Interview_-_Loebner_2013_winner

¹⁶<http://www.square-bear.co.uk/mitsuku/chat.htm>



Figure 2.5: Illustration of possible types of information that Mitsuku can remember from conversations

tion is available (at the time of writing) online¹⁷, where the method is described in a chatbot tutorial¹⁸. Their agent generates responses by using a transformer model. A persona and conversation history are considered when generating each response [51].

A short interaction with the demo reveals the agent does illustrate a persona, however it suffers from severe repetition.

Short interaction example

User: hello, how are you today?
Machine: i am good, just playing my computer games, you?
User: which games?
Machine: i play a lot of video games, i love them
User: what is your favorite game?
Machine: i'm a game person, i love playing games
User: what else do you like doing?
Machine: i like to watch the world series
User: What genre of games is your favorite?
Machine: i like to watch the world series.
User: that is cool. what is your favorite game?
Machine: i love to play video games.

2.4.4 Microsoft's XiaoIce

XiaoIce is a social chatbot developed at Microsoft, which is based on a hybrid AI engine that combines rule-based and corpus-based (or "data-driven") methods [59]. It was designed as an AI companion that considers and recognizes human feelings for the purpose of establishing long-term relationships with human users. The team behind XiaoIce has developed over

¹⁷<https://convai.huggingface.co/>

¹⁸<https://medium.com/huggingface/how-to-build-a-state-of-the-art-conversational-ai-with-transfer-learning-2d818ac26313>

230 skills for the agent, consisting of both non-task-oriented and task-oriented skills such as question answering, recommending movies or restaurants, storytelling, ticket-booking, etc. Additionally, the chatbot features image commenting where given an image by a user, the agent will comment on the image, and content creation such as generating poetry together with the human users.

The part that is most related to this thesis is their "Core Chat" component, which handles open-domain conversations with users. The component is described as a data-driven response generation system. It consists of two retrieval models, one generative model, an answer ranker, as well as editorial responses. The Core Chat component is combined with a component called "Empathetic Computing". It calculates a dialogue state vector consisting of a contextual query, conversation context, query and response empathy vectors. The conversation context consists of a number of the previous messages in the conversation. The contextual query is created by rewriting the user query by using information from the conversation context. The empathy vector is a vector that is encoded with the user's feelings and states in the conversation. The response empathy vector encodes an empathetic aspect, based on the agent's personality and the current situation with the user, that is expected in the agent's next response.

The first retrieval model is stored with paired data, consisting of query-response pairs. These pairs are collected from two sources, the internet (social media, forums, news comments, etc.), and from the human-machine conversations generated when users interacted with XiaoIce. It is estimated that 70% of the agent's responses nowadays are retrieved from the agent's past conversations. The data collected from the internet is filtered by converting each query-response pair into a tuple, consisting of the contextual query, response as well as the user and response empathy vectors, by using the empathetic computing module. The data is filtered based on the tuples to only retain data pairs that contain empathetic responses that fit XiaoIce's persona. Retrieval structures cannot cover all topics (especially new topics unless the database is regularly maintained).

A test was performed using three models, a retrieval-based model, a neural-generator-based and a hybrid of the two models. During testing, the retrieval-based model achieved the second highest score, and the neural-generator-based model achieved the lowest score. It was found that the retrieval-generative hybrid model achieved higher score compared to the two other models by themselves [59]. Therefore, a neural response generator was added to the Core Chat component, to turn it into a retrieval-generative hybrid component. The generator was trained using the paired data from the retrieval database. The model was based on the sequence-to-sequence (seq2seq) framework [44]. A variation of a persona-based model, which is a type of seq2seq model, was used for XiaoIce's neural response generation to reduce the occurrences of inconsistencies in outputs, such as an inconsistent persona [24]. The generator takes the contextual query and a linear combination of the empathy vectors, as input to generate responses that consider the context, the user profile, and the agent's persona. Beam search is used to generate up to 20 response candidates.

An additional retrieval model that stored "unpaired" data, was introduced as complementary to the other two models, to further improve the coverage of the agent's responses. Although the neural response generator is able to provide a response to any topic, the responses are shorter and contain less useful content in regard to the topic compared to the response candidates from the unpaired database. Its database consists of sentences collected from public lectures and quotes in news articles and reports. The data is yet again filtered to align with the agent's persona. The unpaired database should not be used by itself, or else it risks retrieving responses that repeat what the user just said. Therefore, a knowledge graph was constructed which contains triples of the form head-relation-tail. Each triple contains a pair of related topics (head, tail) that people often discuss in one conversation, e.g. (Einstein, Relativity) or (Quantum Physics, Schrodinger's cat). Then, given a contextual query, a topic is extracted from the query and a related topic is found through the knowledge graph, and thereafter a query can be made to the database with the two topics.

A ranker using a 'bosteed tree ranker' ranks the candidate responses from the three models. A pre-set threshold is used to classify whether the candidate responses are acceptable, determined by if the candidates' ranking scores are above the threshold. A response is then chosen by randomly sampling from the acceptable candidates. The ranker calculates scores based on four categories of features.

1. Local cohesion features, how related is the candidate response to the contextual query?
2. Global coherence features, how coherent is the candidate response with the conversation context and the contextual query?
3. Empathy matching features, how well does the candidate response match the personality of the agent?
4. Retrieval matching features, for responses from the paired database, how well does the query in the query-response pair match the contextual query?

If no valid response is retrieved or generated, then an editorial response is given. Examples of editorial responses that XiaoIce may respond with are "Hmmm, difficult to say. What do you think?" or "Let us talk about something else."

Although the chatbot sound ideal when reading through the report, as it considers both a persona and context of the conversation, investigation of interactions with the agent show a lack of context understanding¹⁹. It attempts to mask this lack of understanding by changing the topic. The research paper does express the need for breakthroughs in memory modeling, as well as many other areas, for chatbots to achieve human level intelligence [59].

2.4.5 Meena

Meena is a chatbot created by Google [1]. The model is an end-to-end trained neural conversational model with 2.6 billion parameters. It uses a type of Transformer architecture called The Evolved Transformer [38] and was trained on 341GB of filtered social media conversations. Compared to the largest GPT-2 model, Meena has 1.7x greater model capacity and was trained on 8.5x more data. The trained chatbot was presented as being more sensible and specific than existing state-of-the-art chatbots, as backed up by crowd-sourced human evaluation. Their user test was done by collecting about 100 conversations for well-known open domain chatbots: Mitsuku, Cleverbot, XiaoIce [59] and DialoGPT [57]. Conversations between humans and Meena, Mitsuku and XiaoIce are available on Google's GitHub²⁰. For consistency, each conversation started with the same greeting from the user "Hi!". The crowd workers answered two questions for each utterance from the chatbot, "does it make sense?" and "is it specific?", to measure if the response makes sense in context and whether the response is specific to the conversation or a general response. Using the test data, an evaluation metric was created by using the average of the fraction of responses that are marked "specific" and "sensible". Meena scored the highest and XiaoIce scored the lowest²¹. Although, in the report it was noted that due to how the researchers performed the test of XiaoIce, the model's user profiling suffered as well as its context state.

From looking at sample conversations between Meena and human users, it can be observed that the chatbot has impressive context understanding in comparison to other chatbots, but still illustrate flaws. Two flaws are that of repetition and occasionally interacting as if the chatbot's previous response was from the user. Furthermore, the work focused on sensibleness and specificity, and as such does not consider long-term memory or persona.

¹⁹<https://github.com/google-research/google-research/blob/master/meena/xiaoice.txt>

²⁰<https://github.com/google-research/google-research/tree/master/meena>

²¹<https://ai.googleblog.com/2020/01/towards-conversational-agent-that-can.html>

Repetitive Example - Meena Conversation 92

Human: Hi!
 Meena: Hello! What are you up to today?
 Human: Do you like sushi?
 Meena: I love sushi! What kind of rolls do you enjoy?
 Human: I like tuna roll of course
 Meena: Tuna Rolls are the best! How do you feel about spicy tuna rolls?
 Human: They're great
 Meena: Spicy tuna rolls are the best! What kind of rolls do you like best?
 Human: I already told you. What do you like?
 Meena: I like spicy tuna rolls! Do you like spicy tuna rolls?
 Human: do you like any other rolls?
 Meena: I like spicy tuna rolls and spicy salmon rolls! Do you like spicy salmon rolls?
 Human: yeah. Salmon is so tasty
 Meena: What's your favorite kind of roll?
 Human: I think my favorite actually is the Salmon roll
 Meena: What about spicy tuna rolls?

Inconsistency Example - Meena Conversation 1

Human: Hi!
 Meena: Hey there! What's up?
 Human: Same old, Monday blues...what are you up to?
 Meena: I'm procrastinating on a paper I'm supposed to be writing.
 Human: are you thinking of pulling an all nighter?
 Meena: Possibly. What's your paper on?
 ...

Context Understanding Example - Meena Conversation 83

...
 Human: What'd you do today?
 Meena: Stayed home and played video games.
 Human: Oh nice! What games?
 Meena: Dying Light, Borderlands 2, and Bioshock Infinite.
 Human: Oh interesting. Which is your fav?
 Meena: Dying Light, hands down.
 Human: Why?
 Meena: It's just so fun.
 Human: Yeah but like what about it? Is it an FPS?
 Meena: It's like Mirror's Edge, only in a zombie apocalypse.
 Human: Oh that's kinda cool!
 Meena: It's hella fun.
 ...

2.4.6 Replika

Replika is a chitchat system that, like XiaoIce, combines neural generation and retrieval-based methods. Important to note is that there is not a research paper about the system as a whole, and most information is found from slides at the developer's GitHub²². The neural generation considers persona-based embeddings as proposed in [24] and emotional embeddings as proposed in [58], when generating responses²³. The retrieval-based model is trained based

²²<https://github.com/lukalabs/replika-research>

²³https://github.com/lukalabs/replika-research/blob/master/scai2017/replika_ai.pdf

on a hard negative mining approach, to force their model to produce low matching scores for similar contexts and responses. This is to avoid "echo-responses", where a retrieval-based model retrieves the most semantically similar response rather than the most appropriate [13]. The system also considers conversation history and context by encoding it and providing it to the retrieval and/or generative models. The generative model used was at one point a Hierarchical Recurrent Encoder-Decoder (HRED) [40] which is an extension to Seq2Seq that considers context. Although, in an open source repository²⁴, the developers suggest using Transformer based models instead. Additionally, the agent has the ability to comment on and ask questions about images sent by the user.

Although no official evaluation results have been published on the system measuring its engagingness, etc., the agent is available online to interact with²⁵. The system has a built-in feedback mechanism which allows user to give a thumbs up or thumbs down on any response provided by the agent. With the help of this feedback, a reranking component was then added to the system by training a BERT transformer model to classify whether a sentence would lead to a thumbs up or a thumbs down²⁶.

2.5 User Testing and Evaluation

Throughout the years, work has been done toward creating standardized measurements tools in the field of human robot interaction and dialogue agents. Three examples of this are PARADISE [47], Godspeed [2] and SASSI [17].

2.5.1 PARADISE

PARADISE is a general framework for evaluating and comparing the performance of spoken dialogue agents [47]. The application of the framework is for task oriented and spoken dialogue agents, which differs from the non-task-oriented chat-based dialogue agents as focused on in this thesis. Considering both agent types are dialogue agents, there may exist some evaluation methods that are useful for both of these agent types. In the case studies for the paper, results were collected by three means: recordings of the user interaction, logging various system information, and a survey. The recordings were used to transcribe user utterances, measure the elapsed time and to collect additional information such as whether there occurred any delay for the agent's responses. Logging was used to log what decisions the system made at each state as well as additional information. Example of additional logged information was automatic speech recognition (ASR), where the user utterance was saved from the perspective of the agent and compared to with the actual, transcribed user utterances, to measure accuracy of the recognition. A survey was used to measure text-to-speech performance, ASR performance, task ease, interaction pace, user expertise, expected behavior, comparable interface and future use. Most of the questions used a 1-5 scoring metric from "almost never" to "almost always". Some questions had yes/no/maybe responses. The questions were as follows, where ELVIS is an agent for accessing email over the phone.

- Was ELVIS easy to understand in this conversation? (TTS performance)
- In this conversation, did ELVIS understand what you said? (ASR performance)
- In this conversation, was it easy to find the message you wanted? (Task Ease)
- Was the pace of interaction with ELVIS appropriate in this conversation? (Interaction Pace)

²⁴<https://github.com/lukalabs/cakechat>

²⁵replika.ai

²⁶https://github.com/lukalabs/replika-research/blob/master/scail2019/replika_scail19.pdf

- In this conversation, did you know what you could say at each point of the dialogue? (User expertise)
- How often was ELVIS sluggish and slow to reply to you in this conversation? (Expected behavior)
- In this conversation, how did ELVIS's voice interface compare to the touch tune interface to voice mail? (Comparable interface)
- From your current experience with using ELVIS to get your mail, do you think you'd use ELVIS regularly to access your mail when you are away from your desk? (Future use)

The same three means of results collection can be applied to the user tests in this thesis as well. Instead of audio recordings, message history is stored with timestamps. Similarly, logging can be used to store system state information for each message/action to later be analyzed and compared to the message history. Finally, a survey will be used to collect the test user's perception of the agent.

2.5.2 Godspeed

Godspeed is a series of questionnaires to measure the users' perception of robots. The questionnaires cover five key concepts in human-robot interaction: anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety [2]. Anthropomorphism refers to the attribution of human behavior, emotions or characteristics to non-human entities such as robots or animals. It is used to measure how human the user perceives the agent. Animacy measures life-likeness and perceived safety measures the user's emotional state. Intuitively, likeability and perceived intelligence measures how likeable and how intelligent the user finds the agent. The questionnaires consist of a number of adjectives paired with their corresponding antonyms. Users are then instructed to rate each pair from 1 to 5. An example pair is Fake/Natural, where 1 means fake and 5 means natural. A few more examples of adjective pairs from the questionnaires are Machinelike/Humanlike, Unfriendly/Friendly, Unintelligent/Intelligent and Agitated/Calm.

Out of the five concepts, animacy and perceived safety are not as relevant in the case of chatbots, as they may be for robots. Anthropomorphism is a useful concept to measure perceived realness, and likeability may correlate with engagingness. Perceived intelligence may be associated with realness, where a more intelligent system seems more human-like. However, as there are intelligent and unintelligent humans, designing a chatbot to be human-like should not necessarily mean an intelligent one. As such, anthropomorphism and likeability are considered to some degree when evaluating the chatbots in this thesis.

2.5.3 SASSI

The presence of six main factors in user's perceptions of speech systems was found in the work by Hone. and Graham. where they aimed to create a tool for "the Subjective Assessment of Speech System Interfaces" (SASSI) [17]. The six factors were found after performing principal component analysis (PCA) on a number of questionnaire statements (or items). The six factors were named "System Response Accuracy", "Likeability", "Cognitive Demand", "Annoyance", "Habitability" and "Speed". System Response Accuracy was the name give for questionnaire statements related to the system's accuracy, reliability, predictability, etc. Three example statements are "The system is accurate", "The system is unreliable" and "The interaction with the system is unpredictable". Three example statements for likeability are "The system is useful", "The system is pleasant", "The system is friendly". The name Cognitive Demand relates to the user's emotional state and the perceived difficulty level of using the system, "I felt calm using the system", "A high level of concentration is required when using the

system". Annoyance refers to perceived repetitiveness, engaging-ness and how frustrating the interaction with the system is, "The interaction with the system is repetitive", "The interaction with the system is boring", "The interaction with the system is irritating". Habitability measures how confidently the user perceived themselves to be using and understanding the system correctly, "I always knew what to say to the system", "I was not always sure what the system was doing". The sixth factor only contained two questionnaire statements, "The interaction with the system is fast" and "The system responds too slowly", and was thereby named Speed.

Not all of the six factors are directly applicable to non-task-oriented chatbots. System response accuracy, habitability and cognitive demand may be more applicable to task-oriented agents. Likeability, annoyance and speed are relevant factors for chatbots to measure engagingness. The engagingness as a whole may be affected by how pleasant the interaction with the chatbot is (likeability), perceived repetitiveness (annoyance) and system response time (speed). A system that oftentimes repeats itself and takes a long time to respond to input may not be very engaging to chat with.

2.5.4 Automatic Evaluation of Responses

Evaluating responses of dialogue systems such as chatbots, without human judgement, is a difficult task. Attempts have been made to use similar metrics as seen for machine translation models, such as the metric BLEU [30]. The problem is that these metrics assume that "valid" responses have significant word overlap with the ground truth responses, which is not necessarily the case for responses in the dialogue task [25].

Example of valid response without word overlap

Question: What are your hobbies?

Ground truth: My hobbies include biking and hiking.

Generated response: I like to draw and play video games.

The question in the example may have many different answers which do not share any word-overlap with the ground truth response, which would lead to a low BLEU score, e.g. 0, but the answers are clearly valid.

A survey was performed to determine the correlation between human judgements and automated metrics, such as word-overlap (BLEU) and word embedding (word2vec [28]) methods, respectively. The result was that all metrics show weak or no correlation with human judgement, and a comparison was made that the metrics were similar to random noise [25]. Despite the results of the survey and the poor performance of both metric types, the researchers believed that distributed sentence representations (Word2vec / word embeddings) hold the most promise for the future. As such, automatic metrics will be tested in this thesis as a complement to manual, human evaluation.

2.5.5 Conversation-Turns per Session

To evaluate social chatbots on their engagingness, especially for long-term engagement, a metric was proposed as conversation-turns per session (CPS) [59]. Chatbots are then considered more engaging the larger the CPS is. The metric is calculated as the average number of conversation-turns between the chatbot and the user in one session. XiaoIce achieved an average CPS of 23, by averaging the CPS of conversations collected from millions of active users over a longer period of time (1-6 months) [59].

2.6 Open Source Conversational AI, Rasa

Rasa²⁷ is an open source machine learning framework, which can be a suitable tool for the aim of this project. It provides infrastructure and tools to build contextual assistants that can automate text-and voice-based conversations. The application can be deployed on various platforms such as Facebook Messenger, Slack, Telegram and so on.

Rasa provides an open source version which is sufficient. It contains two components: NLU (Natural Language Understanding), which determines what the user wants and captures key contextual information; and Core, which selects the next best response and/or action based on conversation history.

Additionally, Rasa also provides a free closed source toolset Rasa X, which is a layer on top of Rasa Open Source and can be deployed on a virtual machine hosted on, for example Google Cloud. Rasa X can be used for improving the agent through conversation-driven development (CDD) and allows continuous integration and continuous deployment (CI/CD).

2.6.1 Rasa NLU

NLU stands for Natural Language Understanding, it turns user messages into structured data. With Rasa NLU, this is done by providing training examples which show how the chatbot should understand the user messages, then a model is trained by showing it those examples. Below is a sample of NLU data.

A sample NLU data

```
## intent: greet
- hey
- hello
- hi
- good morning
- good evening
- hey there
- ...
```

As the chatbot understands the user messages, it is able to classify and extract the intents and the entities. A user's intent represents the goal or meaning of the input. For example, the message "Hello" can be identified as a "greet" intent, because the meaning of this message is a greeting. Entity is a keyword that the agent takes consideration of. For example, the message "My name is John" contains information related to a personal name, the agent should then extract an entity, in this case labeled "name" with value "John", and remember it throughout the conversation to keep the interaction natural.

Incoming messages are processed by a sequence of components in Rasa. These components are executed one after another in a pipeline. A pipeline commonly contains three main components:

- Tokenization.
- Featurization.
- Entity Recognition, Intent Classification and Response Selectors.

2.6.1.1 Tokenization

A tokenizer segments text into words, punctuations marks and so on, it splits a text into tokens. There are different tokenizers that can be used for different types of user inputs.

²⁷<https://github.com/RasaHQ/rasa>

For example, Whitespace Tokenizer can be used to process words are separated by spaces, which is the case for English and many other common languages. Other tokenizers are also supported if the language is not whitespace-tokenized.

2.6.1.2 Featurization

A featurizers creates a vector representation of user message and/or response. There are two different types of text featurizers: sparse featurizers and dense featurizers. Sparse featurizers return feature vectors with a lot of missing values, e.g. zeros. These feature vectors usually take up a lot of memory, therefore they are stored as sparse features. Sparse features only store the values that are non zero and also their positions in the vector.

2.6.1.3 Entity Recognition, Intent Classification and Response Selector

Entity extractors extract entities from the user input, such as person names, locations and other topics. Intent classifiers assign one (or more) of the pre-defined intents to the user input. Response Selectors predict a response from a set of candidate responses.

2.6.2 Rasa Core

Rasa Core is responsible for dialogue management, it decides how the chatbot should respond. It learns from supplied conversation examples, i.e. how a conversation between a user and an agent would take place. These examples are called *stories*. An instance of a story contains the user's intent, and the reply(s) that should be triggered.

The replies are supplied as templates, these are called *responses*, which the agent can use. Finally, an environment is defined where the agent will operate, this is called a *domain*.

A domain is essentially the universe that the agent lives in. Within this domain, all necessary intents, entities, and actions that the agent is able to use are supplied.

Additionally, the bot has memory *slots* which store certain information that the user has provided. However, the slots are cleared after each interaction session, in order to store and use information, an external dataframe is needed.

2.6.2.1 Story

The training data in Rasa are given as a form of stories, these are used to train the dialogue management models. A story essentially represents a dialogue flow of a user and the chatbot, it contains both the user's intents (and entities if applicable) and the respective responses and/or actions that the chatbot should take. Below is an example of a simple story that represents the beginning of a conversation. The user's intents are labeled with an asterisk, and the chatbot's responses and actions are labeled with a dash, and responses always start with the "utter_" prefix. Figure 2.6 shows the story flow, the graph can get more complex as the story gets more routes and contains different responses and actions.

A sample story used as training data

```
## example story 1
* greet
- utter_greet
- utter_ask_name
* give_name
- utter_acknowledge
- action_remember_name
```

As seen in the example training data above, an advantage with this format is that the specific input of the user does not need to be present. Instead, the output intents and entities

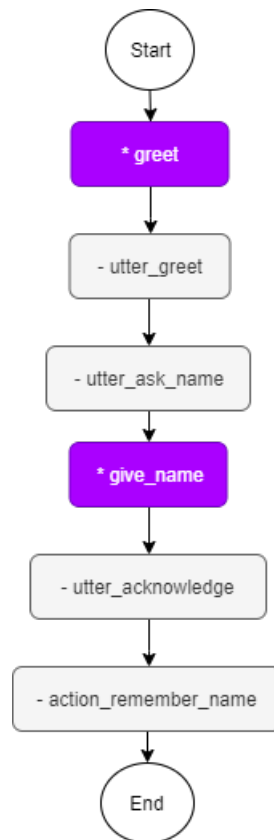


Figure 2.6: A story flow diagram visualized in Rasa X.

from the NLU pipeline are utilized. Therefore, for any messages that are classified as *greet*, the chatbot will take the responses from *utter_greet_back* and *utter_ask_name* and reply. All the available responses are provided in a domain file.

2.6.2.2 Domain

The domain essentially defines the universe that the chatbot lives in. It specifies all the intents and entities that the chatbot should know about, and all the responses and actions that can be used. In addition, Rasa also support displaying images and using buttons, which is a nice touch if the application is to be deployed on other platforms such as Facebook Messenger. Below is a sample of the domain file that contains some intents and entities.

A sample of the domain file

```

intents:
- greet
- goodbye
- affirm
- deny
- mood_great
entities:
- name
  
```

2.6.2.3 Slot

The chatbot is equipped with memory slots, it is essentially a key-value database. A slot holds a certain information such as a personal name or a location, that is provided by the user during the interaction. It is also possible to retrieve information from other sources, for example query from database.

Slot has various types for storing different kinds of information, for different behaviors. Below is a list of supported slot types.

Supported slot types

- Text
- Boolean
- Categorical
- Float
- List
- Unfeaturized

Situations where the value does not matter, for example when discussing weather, then a *text* slot named "location" can be used. When the user provides their current location, this value is then stored into the slot, the exact location does not matter to Rasa Core, whether it being in New York or Paris. Rasa Core only needs to know that a location entity was given, and its value will be stored in this specific slot named "location". *Unfeaturized* type is another slot type where the value does not matter.

If the conversation flow or responses will depend on the value, then there are other types of slot that can be used, including *bool*, *categorical*, *float* and *list*. For example, the chatbot asks a boolean question, and it stores the answer to a *bool* type slot, later on the chatbot's responses can vary depending on the boolean value.

2.6.2.4 Response

After the chatbot has understood what the user is saying, it needs to reply accordingly. In order for the chatbot to reply, it needs to be supplied with one or more responses. These responses are also defined in the domain file, and are used in the stories. Below are a few examples of responses that the chatbot can use.

A few examples of responses

```
utter_greet:
- text: "Hey! How are you?"
utter_bye:
- text: "See you next time!"
```

One or more responses are used given one or more certain intents. Below is a sample of training data written in the story format.

An example of story that is used as training data

```
## story path 1
* greet
- utter_greet
* mood_great
- utter_happy
```


2.6.3 Rasa X

Rasa X is a free but closed source toolset that can be used to improve the agent by deploying the agent on a server and let users interact with it, gather additional information which can be used for example as more training data, or to correct false intent and entity classification manually and so on. It comes with a neat UI and is easy to navigate and share the agent. Rasa X is deployed using Docker, which is a tool that uses containers to make the process of creating, deploying and running applications easier.

2.7 VADER Sentiment Analysis

An external component, a sentiment analyzer from the NLTK (Natural Language Toolkit), specifically the VADER (Valence Aware Dictionary and sEntiment Reasoner)²⁸, is considered to be used as a ranking system and be included in the pipeline. This component mainly returns a sentiment value, either “POSITIVE”, “Neutral” or “NEGATIVE”, along with its confidence score. Figure 2.7 shows a few examples of different user input and the respective sentiment analysis results.

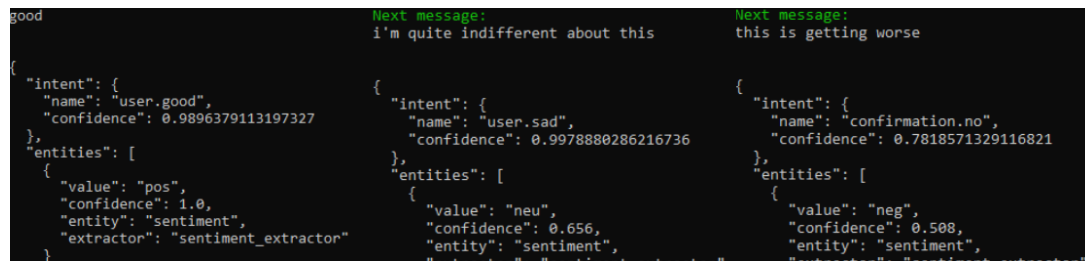


Figure 2.7: Results from the sentiment analyser for different user inputs.

The purpose of the ranking system is to sort the extracted entities by their sentiment values, and the chatbot will prioritize to generate questions related to the topics that have higher sentiment values.

2.8 Semantic Network

A semantic network such as ConceptNet [41] is designed to help computers understand the meanings of words that people use. It is considered to retrieve information related to the topics input from the users by calling its API, which returns a list of relations. This search term is the extracted entity from the user’s input. Figure 2.8 shows parts of the relations that ConceptNet provides.

2.9 Conclusion

XiaoIce illustrated that using a hybrid retrieval and neural generative based approach allowed for a more engaging system, as the two models complement some of each-other’s weaknesses. As the generative model that was used in XiaoIce was based on Seq2Seq, perhaps with a Transformer architecture, such as GPT-2, even more engagingness can be achieved. From the work by [3], it is found that avoiding repetition and introducing a persona for the agent leads to a more engaging system in their case. Therefore, a hybrid rule-based and corpus-based approach was chosen for the chatbot in this thesis, with templates, a retrieval structure and a generative model. Templates will be used together with the agent’s

²⁸<https://github.com/cjhutto/vaderSentiment>

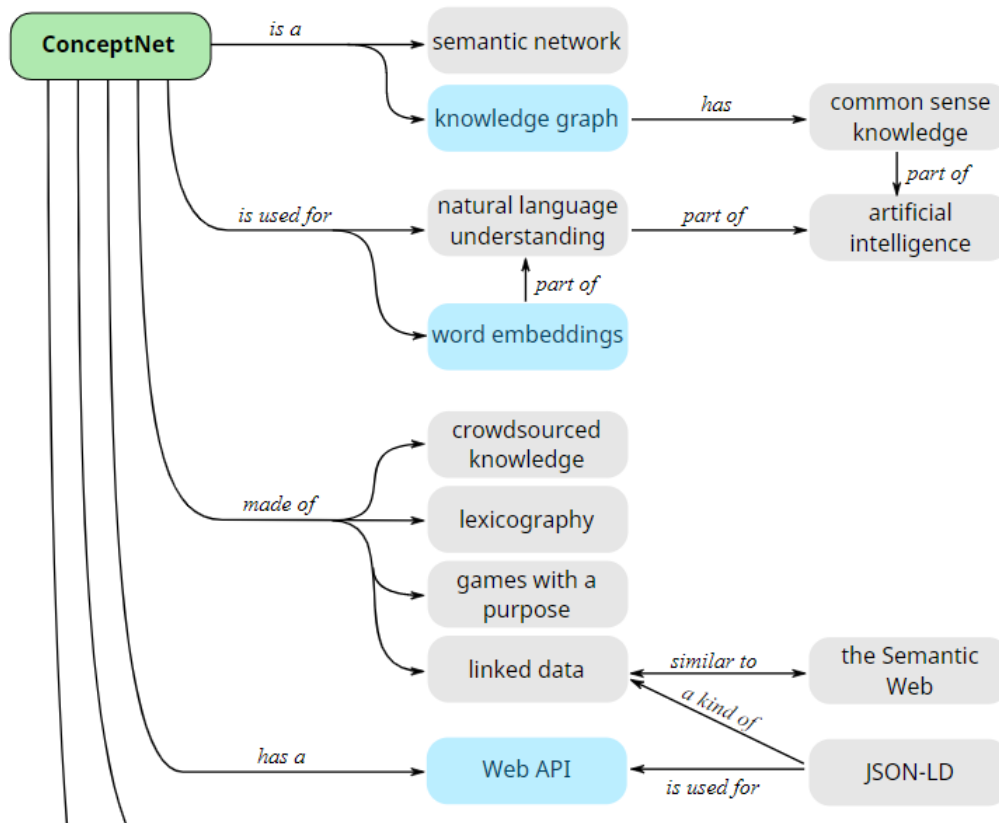


Figure 2.8: Some relations provided by ConceptNet.

personal memory, where a number of variations of the templates are created to reduce repetition. A retrieval structure will be made with high quality responses for typical questions asked when getting to know someone. The generative model will use a GPT-2 model, fine-tuned on the question answering task.

Rasa Open Source provides infrastructure and tools necessary for building high-performing, resilient and contextual chatbot. The ability to understand messages and use machine learning to improve those conversations, integrate chatbots with existing systems and channels, all in one package. Rasa's NLU provides the technology to understand messages, determine intent, and extract key entity. It has a wider usage with supporting multiple languages, and both pre-trained and custom entities. With a very first basic chatbot built, it is then possible to move forward quickly by learning interactively from real conversations with the users. Options for deployment include on-prem, private cloud, or third-party cloud provider. Connecting to existing systems, API calls and knowledge bases, and builds on website, apps, messaging apps or custom channel. It is then decided to study and adapt to the framework.

Human evaluation was found to be the most reliable form of evaluation, compared to existing automatic metrics, which is why user tests will be the main method of evaluation. A decision was made to not use any one of the standard evaluations (PARADISE, Godspeed, SASSI) but rather to take inspiration from them when creating the user surveys for the dialogue agents of this thesis. The reason being, neither of these three standards were specifically developed for measuring users' perception of chat-based dialogue agents.

In both SASSI and Godspeed, it can be observed that several questions are used to measure the same type of aspects. For example, SASSI has the statements "The interaction with the system is fast" and "The system responds too slowly", which both measure speed. This

may offer more robustness as users may interpret questions differently based on the wording of the sentences. Therefore, this concept will be considered when designing the surveys in this thesis.

The evaluation metric of CPS will not be used, purely due to the limited scope of the thesis. The limitation being, that there is not enough time for the development of a fully functional chatbot for long term usage, such that the metric can be utilized. The evaluation will be based on survey results from test users that interact with the agent.

3 | Development of a Question Answering Chatbot

This chapter describes the development of the question answering chatbot followed by the first user test (3.10). A brief overview of the chatbot and its architecture is presented in section (3.1), and then discussed in detail in the sections that follow. The chatbot consists of three main components, a generative model (3.3), a template structure (3.5) with related sentiment memory (3.6) and a retrieval structure (3.7). The components are first introduced by themselves and then put together (3.8). The creation of each component is based around conversational datasets (3.2) that were preprocessed (3.3.1) and analyzed (3.4).

3.1 Architecture

The architecture for the question answering chatbot can be seen in figure 3.1. The figure depicts an initial overview as to how the chatbot's components work together. The chatbot consists of three components: a generative component, a template component and a retrieval component. Furthermore, a front-end was developed to allow users to interact with the chatbot online. The chatbot runs locally on a computer and can be accessed by using flask, a web application framework in python, combined by using a Ngrok tunnel to redirect users from Ngrok to the computer running the chatbot. Then the user is able to send and receive messages to/from the agent. When the user sends a message, the message is sent to the template and retrieval components. The template component attempts to extract any existing nouns, if one exists then the message is transformed into a question template form. The transformed message is then compared with existing template questions to find the most similar template based on a cosine similarity score. Then the retrieval component finds the most similar question in its retrieval database. If the template question has a higher similarity than the retrieved question, to the user message, then the template is chosen and the related answer template is fetched. Otherwise, the retrieval answer is chosen. If the chosen answer has a higher cosine similarity than a pre-defined threshold, then the answer is sent out to the user. Otherwise, an answer is generated and then sent out to the user. The memory of the template and retrieval structures are not explicitly illustrated, but it can be considered part of the respective components.

The following sections describe more in-depth the method applied to develop each individual component, as well as essential datasets.

3.2 Datasets

To train a neural model on a specific task, it is essential to have relevant data for the task. As such, conversational datasets were used to train generative models to generate answers to questions, and to create a question-answer pair retrieval database. Additionally, conversational datasets were analyzed to create templates based on common sentence structures found in questions.

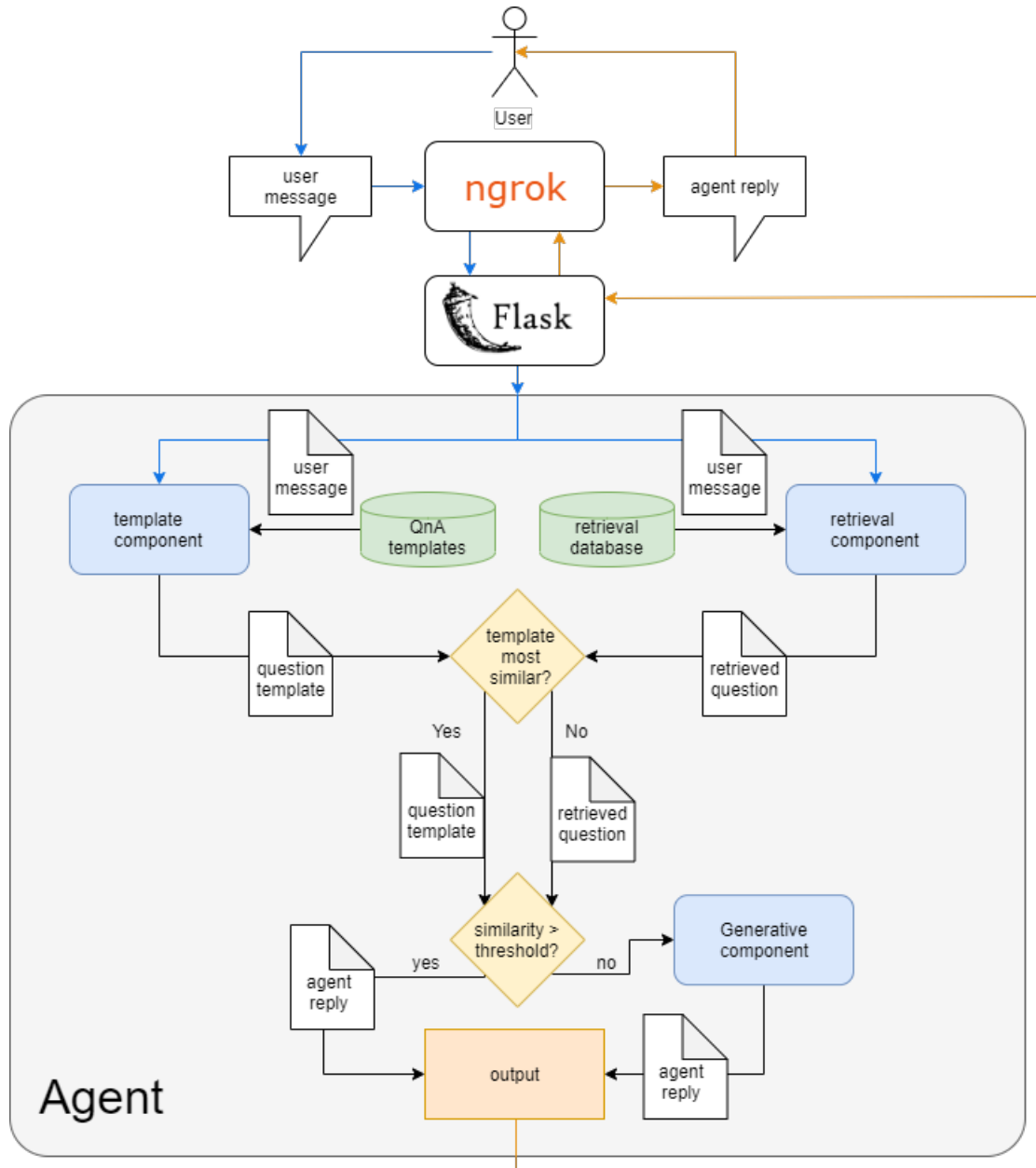


Figure 3.1: System flow diagram describing how input data flows through the architecture of the chatbot and generates output

Four existing datasets were used during this thesis, Cornell Movie–Dialogs Corpus (CMDC) [10], Persona-Chat [56], ConvAI2 [12] and a “getting to know you” (referred to as GTKY from now on) dataset from a study by Huang et al [18]. All datasets except for the ConvAI2 dataset were at one point used for the purpose of training machine learning models. The GTKY and the ConvAI2 datasets were used to analyze what type of questions are commonly asked in a casual chit-chat setting, and what topics occurred. Analyzing these datasets is important for the creation of question and answer templates, see section (3.5). The ConvAI2 dataset consists of human-to-bot conversational data. Analyzing this data may be useful to get a better understanding of how humans interact with agents. The other datasets are human-to-human datasets, where Persona-Chat and the GTKY dataset are messages gath-

ered from a 1-on-1 chat room environment. The Cornell Movie–Dialogs Corpus on the other hand consist of conversations extracted from raw movie scripts [10]. The Persona-Chat and ConvAI2 datasets are related to the same task. The Persona-Chat dataset consists of conversations between two randomly paired crowdworkers, who were asked to act out a given persona and to get to know each other [56]. The ConvAI2 dataset is the result of a competition where the competitors used the Persona-Chat dataset to try and create the best chatbot that acts out a persona [12]. For the GTKY dataset, participants were told to chat for 15 minutes with the objective to get to know each other and learn about each other’s interests [18].

3.3 Generative Model

GPT-2 achieving state-of-the-art results on several language modeling datasets made the model an attractive solution for generating answers in this thesis. Furthermore, GPT-2 was able to generate text faster than other auto-regressive models that were briefly tested (2.2.4.3), which is essential for real-time interaction. A python library called `gpt-2-simple`¹ was used for fine-tuning the GPT-2 models, and for text generation. The maintainer of the `gpt-2-simple` library, Max Woolf, has also provided a Colaboratory notebook² illustrating how to use the library. Google Colaboratory was used to fine-tune different models since they provide a Tesla T4 GPU with 16GB memory (or similar), in this case to fine-tune the smaller models (124M/355M). The larger models (774M/1558M) can also be trained by changing the run-time type from GPU to TPU and “requesting” more RAM from Google Colaboratory, upon which 35.35GB of RAM may be lent. When using the free version of Google Colaboratory, there is currently no guarantee what GPU and how much RAM is available³.

The 124M and the 355M models were the models that were chosen to be investigated further. The main reason was that the time it takes to generate text needs to be as low as possible if the model is to be used in a chatbot system where users may expect a reply within a few seconds. Additionally, the larger models require not only longer time to be fine-tuned but they also require more VRAM (both during fine-tuning and text generation). A short experiment was conducted to compare the models in terms of how long it takes to fine-tune as well as how long time it takes to generate text (A.3). Eventually, the 124M model was chosen mainly due to the faster text generative capabilities compared to the larger models, such that users get their reply from the chatbot as fast as possible. As an example, the 124M model is over twice as fast as the 355M model.

3.3.1 Preprocessing and Creating New Datasets

OpenAI trained the GPT-2 model on 40GB of various types of texts. Therefore, the released model is already able to generate text and can be used for different natural language processing tasks by fine-tuning the model for a specific task (2.2.2). As such, it is necessary to preprocess conversational data to be able to fine-tune the model for the task of question answering.

Existing code for preprocessing the Cornell Movie–Dialogs Corpus was used to create “message pairs”⁴. For example, given three messages in the chronological order x , y and z , the result would be two message pairs (x, y) and (y, z) . This would lead to training data that could train a model to produce both message y if given message x , or message z if given message y . The movie dialogs were thereafter preprocessed, and a new dataset was created; MD-pairs. To train a GPT-2 model on this type of data, it is beneficial to add special tokens at the start and end of a segment. The GPT-2 model is trained with an unsupervised method and

¹<https://pypi.org/project/gpt-2-simple/>

²<https://colab.research.google.com/drive/1VLG8e7YSEwypxU-noRNhsv5dW4NfTGce>

³<https://research.google.com/colaboratory/faq.html>

⁴<https://github.com/floydhub/textutil-preprocess-cornell-movie-corpus>

thus, the tokens help the model to learn a desired pattern. The tokens were called ‘<|startof-text|>’ and ‘<|endof-text|>’. A new dataset was created; MD-token-pairs, by adding a SOT (start of text) token at the start of a message pair, and an EOT (end of text) token at the end of the message pair. This entails that the GPT-2 model can learn to recognize the structure of a message pair.

To be able to train a machine learning model on the task of answering questions, it is necessary to perform preprocessing of data into questions and answers. This will entail that when the model is given a question as input, it will generate an answer as output. The preprocessing code was altered to extract questions and answer pairs instead. The python library NLTK was used to split a message into sentences, and for each sentence, split into tokens. If the last sentence in the message ends with a question mark, then the message is classified as a question. The following message is then classified as an answer. The first question and answer dataset, MD-token-questions-simple, that was used for fine-tuning had the condition to extract the last question in the current message and the first sentence with a punctuation in the following message. This condition was made because there exist messages with several questions. Therefore, the hypothesis was that the last question is the most relevant one. Similarly, it was hypothesized that the first sentence in the following message, that ends with a punctuation, is the most relevant sentence that answers the question. In this case, it was not desired to extract answers that answer a question with a question, as the model is to be a part of a question answering component. As such, the component only handles questions and not answers from the user.

Thereafter, the MD-token-questions dataset was created after adding three conditions to improve the training data. To “improve” the training data here refers to processing the data into a format that is more desirable for this specific task. The question message must consist of more than two words to avoid nonspecific questions like “Why?”, “Why not?”, “Huh?”, etc. These questions were avoided as they are dependent on the previous messages in the conversation; context dependent follow-up questions. The questions were deemed as out of scope at the time due to time constraints of developing the necessary context understanding for answering such questions. Another condition was that the answer must be longer than one word for the sake of avoiding “Yes/No” answers. Ensuring that answers are longer than one word should lead to more engaging generated answers by the model once it is fine-tuned. Finally, the question and the answer must be from two different characters and from the same movie (movie dialog corpus). Different question and answer datasets were created with variations on these conditions where the changing factor was how much of the answer and the question messages were extracted as the question and the answer. For example, MD-token-questions dataset had a condition to only extract the question from the message containing the question, and only the first sentence (that ends with a punctuation) for the answer message. One variation, MD-token-questions-2, used more than one sentence in the answer, to allow the answer to be more complex. Compare the two variations in the following example where MD-token-questions-2 is the “A2” answer.

Q: How do you get your hair to look like that?

A1: Eber’s Deep Conditioner every two days.

A2: Eber’s Deep Conditioner every two days. And I never, ever use a blowdryer without the diffuser attachment.

Although MD-token-questions-2 may result in more engaging answers, it also contains the risk of including sentences that are not related to the question. Therefore, models were mainly fine-tuned using MD-token-questions. Another dataset, MD-token-questions-4 extracts the entire question message and the entire answer message. This results in cases where the answer contains a question as well and may end up being the next question.

Example of MD-token-questions-4 pair extraction

Pair 1

Q: You got something on your mind?

A: I counted on you to help my cause. You and that thug are obviously failing. Aren't we ever going on our date?

Pair 2

Q: I counted on you to help my cause. You and that thug are obviously failing. Aren't we ever going on our date?

A: You have my word. As a gentleman

The GTKY dataset was preprocessed using the same method as when creating the MD-token-questions dataset, henceforth referred to as GTKY-preprocessed. The Persona-Chat dataset was preprocessed into pairs similar to the MD-token-pairs dataset.

3.3.2 Manual Cleaning of Data

The GTKY dataset was not formatted in a way that supported clean automatic extraction of question and answer pairs. This may be because the original data was retrieved from a study in a message chat environment [18]. As such, users may decide to write multiple messages, and to answer a specific message, or part of a message, in arbitrary order.

Example of GTKY data format

User2: How are you today?

User1: fine. how are you?

User2: I am good, enjoying the holidays. Do you celebrate Christmas?

User1: as much as i celebrate any holiday :) | I enjoy cooking a lot in the winter time | so around XXmas time, i start cooking a lot more | especially soups. | do you like to cook?

User2: I have two small children and my 5 year old is really enjoying finding "elf" this year and is anticipating sanat coming to visit. Are you from around here or will be heading home for the holidays? | Yes, but I find I am not doing it as often these days! What is your favorite foods?

User1: this is essentially my home now. no travel for me.

User2: Me as well, I was raised just outside Boston, so I host at our condo in Somerville. Where do you live?

User1: i like soups: kale-white bean-bacon is one of my favs.

"Clean" extraction in this sense refers to extracting the related answer sentence to the specific question. Therefore, after automatically extracting and creating a question and answer pair dataset, GTKY-preprocessed, manual cleaning was done to remove pairs where the answer was not related to the question. Consider the previous example where automatic extraction would give the unrelated question and answer pair: "Q:What is your favorite foods? A:this is essentially my home now.". The manual cleaning reduced the dataset by roughly 44%, from 2035 pairs to 1147. Even though the manual cleaning of the dataset decreased the size of the training data, the current work aimed at improving the model performance based on quality and relevance of the training data rather than the size.

3.3.3 GPT-2 Fine-Tuning

Table 3.1 shows several models that were fine-tuned through this project using the gpt-2-simple python library with its default training parameters (e.g. utilizing Adam optimizer with a learning rate of 1e-4 and a batch size of 1). The first test was done by fine-tuning a 124M parameter model on the MD-pairs dataset, and a 355M parameter model on the MD-token-pairs dataset. The test's purpose was to get familiar with the fine-tuning process. The

models were not saved, and no comparison of the models was performed. Thereafter, a few models were trained on different question and answer datasets, where the datasets were created with different conditions, as mentioned in (3.3.1). Most of the early models were not saved or compared extensively. The main purpose of the early models was to continuously fine-tune models on different datasets, to find what dataset is better at teaching a model the desired behavior of generating personal questions and answers. The output of a model was observed and thereafter a new model was trained to see if there was a noticeable difference or improvements between the current model and the previous model.

After the GTKY-preprocessed dataset was created, and a model (#7), was fine-tuned on the dataset, the model started showing the desired behavior of asking and answering more personal questions. An example of four different models answering personal questions is provided in the appendix A.2. The MD-token-questions size is 3.53MB compared to GTKY-preprocessed dataset's size of 226kB. Fine-tuning 400 iterations on the significantly smaller dataset may have resulted in an over-fitted model. The GTKY-preprocessed dataset was thereafter manually cleaned into a new dataset, GTKY-preprocessed-manual. Three models were trained purely on the GTKY-preprocessed-manual dataset, model #8, #9 and #10. After #8, models with 124M parameters were trained instead of 355M to ensure less risk of overfitting, as the dataset was very small. Model #10 was chosen as the generative model for the system, as it seemed to generate relevant answers more often than the other models.

Model	parameters	fine-tuning description
#1	124M	1000 iterations on MD-pairs dataset.
#2	355M	1000 iterations on MD-token-pairs dataset.
#3	124M	1000 iterations on MD-token-questions-simple dataset.
#4	355M	1000 iterations on MD-token-questions-simple dataset.
#5	355M	1000 iterations on MD-token-questions-2 dataset.
#6	355M	1000 iterations on MD-token-questions dataset.
#7	355M	1000 iterations on MD-token-questions and thereafter 400 iterations on the GTKY-preprocessed dataset.
#8	355M	100 iterations on the GTKY-preprocessed-manual dataset.
#9	124M	100 iterations on the GTKY-preprocessed-manual dataset.
#10	124M	200 iterations on the GTKY-preprocessed-manual dataset.

Table 3.1: Fine-tuning GPT-2 models

3.4 Data Analysis

The GTKY dataset was analyzed to understand what questions were commonly asked when getting to know someone, such that the chatbot can be developed to handle the most common questions. The analysis was done by going through the conversation data and extracting every sentence that ended with a question mark, which resulted in 5167 questions. This was done by using NLTK's sentence tokenizer and word tokenizer to divide a message into sentences and thereafter dividing sentences into tokens. Then, part of speech tagging (POS) was applied to every question to find what nouns most commonly occur in the questions. This was done by using spaCy to tokenize the questions, as it contained a POS tagger (NLTK can also be used).

Nouns were extracted from the questions and added into a dictionary together with the number of occurrences of the noun. Nouns with less than 10 occurrences in the dictionary were disregarded which resulted in 87 different nouns. The assumption was that the higher the number of occurrences for a noun, the more that topic or that type of question which uses the noun is asked. A total of 4955 nouns were extracted (not unique), from which the top 87 nouns with a total of 2653 summed occurrences make up 53.5% of the extracted nouns. The

appendix includes table A.3 where the top 87 nouns are illustrated together with a question that consist of the noun. Thereafter, questions that contained a noun were grouped together in a list, which resulted in 2754 remaining questions compared to 5167.

For each noun, the associated questions were analyzed. The analysis was performed by manually looking at the questions for the specific noun to make an assessment of what type of questions were asked that make use of the noun. For example, the most common noun “time” was mostly associated with the question “What do you like to do in your free time?”. The questions that were perceived to be most commonly asked for each noun were written down in a txt file, to be used when creating a retrieval database and as reference when creating templates. A total of 652 questions were written down. Attention was also put when writing down questions on which questions could be used as template questions by replacing the keyword (or the noun) of the sentence.

All the extracted questions (5167) were finally clustered to determine common topics. To be able to cluster the questions, every question was transformed into an embedding vector, using Universal Sentence Encoder⁵ [6]. The elbow method was used to decide on the number of clusters to use (n). The elbow method is a heuristic which determines an optimal clustering of the data. The method is to plot a metric, such as distortion (y-axis), over the number of clusters (x-axis) to find a point where the rate of improvement per additional cluster decreases. Initially, the metric value will improve steeply (lower value is better) as the number of clusters increases, but at one point the rate of improvement flattens. A lower number of clusters is desired and therefore the number can be decided as a point before the curve flattens. In this case, k-means clustering was called in a for-loop in the range $n=[2, 39]$. In each iteration, the sum of square distances of each sample to the closest cluster center was calculated. Figure 3.2 shows the distortion value changes across different n values.

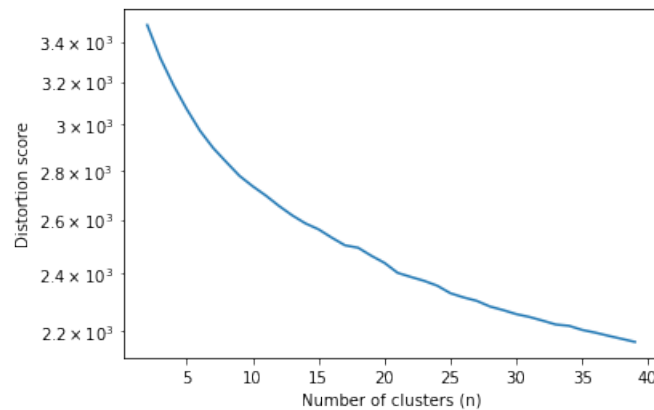


Figure 3.2: Plotting distortion over the number of clusters.

From the figure, there was no obvious number of clusters to choose. Therefore, attempts were made with $n = 10$ and $n = 20$. After analyzing the clusters and what type of questions existed in each cluster, it could then be observed that $n=10$ was too few whereas $n=20$ was too many. Each cluster was analyzed by printing the questions in the cluster and extracting the nouns from the questions. After analyzing the dataset, the most common themes/topics could be abstracted to one of the following (unordered):

- hobbies/interests/fun activities/plans
- personal (age/name/marital status/children/pets/family/residence/country)
- music/instruments

⁵<https://tfhub.dev/google/universal-sentence-encoder/1>

- movies/TV shows
- student/study/research
- weather
- food
- sports
- work/job/money/business
- books
- travel/language/vacation

3.5 Templates

Previous work at University of Twente developed a template-based framework for generating follow-up questions [27]. The framework consisted of three major parts: pre-processing, deconstruction, and construction. Pre-processing to clean-up user input, deconstruction to find a sentence pattern, and construction to construct a response by matching the sentence pattern obtained in the previous step. The template component in this chatbot contains the same three parts but with different methods and for question answering instead of question asking. A main difference is that [27] created rules to find patterns in user inputs which were then linked with a template response, whereas this component developed template questions linked with template answers. By using Semantic Role Labeling [27] is able to detect inputs with the same semantic meaning but that are syntactically different and match the inputs to the same response. Whereas in this component, user inputs were compared to template questions by transforming the input and templates into sentence embeddings and calculating the cosine similarity between the input and the question templates, to detect a matching question.

Recurring sentence structures were observed in some questions during the analysis of the GTKY dataset, which led to the creation of 21 question templates. The 21 templates were then divided into 6 types of questions, by assigning them an answer ID. Answer templates could then be made for a specific question type by assigning the question type's answer ID to the answer template as well. The templates were then used in a retrieval way, where the user's input is matched to a template question (deconstruction) to retrieve a related template answer which is then further processed (construction). Many of the question templates have only a slight variation which is how the 21 templates could be divided into 6 types. Having variations of the same question adds more robustness as it can recognize more inputs. The question templates are shown in A.4.

The 6 template types

Do you <sentiment> <noun>? - Do you like pizza?

What kind of <topic> do you <sentiment>? - What kind of music do you like?

What is your favorite kind of <topic>? - What is your favorite kind of food?

What's your favorite <topic> genre? - What's your favorite music genre?

What do you think about <noun>? - What do you think about cats?

What are some of your favorite <topic>? - What are some of your favorite foods?

A template is retrieved if the user's input is similar enough to the template. The similarity is calculated by turning the templates and the input into sentence embeddings and calculating the cosine similarity. The templates consist of three token types: <sentiment>, <noun> and <topic>, which enables the templates to be used with different topics, subjects (noun), and sentiment. Given a user input, a copy of the input is processed to follow the template

sentence structure. Sentiment keywords such as “like/dislike/love/hate” are replaced by the token ‘<sentiment>’. If the sentence contains a noun, as recognized by POS tagging by spaCy or NLTK (both are used for robustness), then the noun is added into a list of extracted nouns. The pluralized, singularized and lemmatized forms of the noun are also added to the list to be able to use a direct comparison to nouns in the sentiment memory. For each noun in the list, the noun is compared against the subjects and topics in the sentiment memory. The template questions were designed to only consider one noun, and therefore the algorithm stops going through the list after it finds a noun that exists in the sentiment memory. The noun is then replaced by the token <noun> or <topic> depending on if the noun is a subject or a topic, for example:

Q: Do you like bananas? banana -> <noun>

Q: What food do you like? food -> <topic>

It was considered to perform additional analysis to find if there were common sentence structures for answers reoccurring in the dataset when answering these types of questions. In this case however, the answer templates were manually written to fit the question templates. The answer templates are shown in A.5 and are further described in (3.8).

3.6 Sentiment Memory

It is natural for people to like some things and dislike others. One person may love apples but hate oranges, while another person may love oranges and hate apples. A type of memory was constructed to simulate this aspect, such that the agent has a clear sense of preference. The agent should be able to say whether it likes certain things such as apples, and what the agent’s favorite fruit is. Therefore, noun words were manually collected from different categories and stored in a csv file with a randomized value in the range between 0.0 and 1.0. The categories considered in this case are: food, fruits, desserts, sports, hobbies, music (genre), genre (movie/books), books, movies, TV shows, colors and animals. All categories except for colors were decided based on the most common themes/topics found during the previous analysis (3.4). Nouns were manually extracted for each category resulting in a total of 357 nouns, with a minimum of 8 nouns for each category, with food, fruits and desserts consisting of the majority of the nouns (214/357). ConceptNet⁶ was used to find nouns for most categories (food, animals, colors, music (genre), genre (movie/books), sports and hobbies). IMDb⁷ was used to collect a few popular movies and TV shows. Books were collected from a user rated top list found on the internet⁸. Fruits⁹ and desserts¹⁰ were extracted from vocabulary word lists. The random value stored along with the noun determines the sentiment for the noun. Nouns with sentiment values above 0.5 are classified as “like” while values below 0.5 are classified as “dislike”. Choosing a random value entails that the agent’s preferences can be assigned without manual labor. The nouns are stored in the format of (topic, subject, sentiment). This allows the agent to answer specific questions such as “Do you like apples?”, by searching for the noun apples in the csv row “subject” in its memory and giving an answer based on the related sentiment. It also allows the agent to answer more open questions such as “What fruits do you like?” by recognizing that fruits is a category or a “topic”, and thereafter returning an answer with the fruit with the highest sentiment in its memory.

The sentiment memory could be populated dynamically during conversations by using ConceptNet’s API to find the IsA relationship for each new/unknown noun. The IsA relationship classifies nouns into the topics they belong to, for example football -> is a sport,

⁶<http://conceptnet.io/>

⁷<https://www.imdb.com/>

⁸https://www.goodreads.com/list/show/1.Best_Books_Ever

⁹<https://www.enchantedlearning.com/wordlist/fruit.shtml>

¹⁰<https://www.enchantedlearning.com/wordlist/desserts.shtml>

banana -> is a fruit/food. The sentiment memory could therefore grow larger and larger the more users interacted with the system by saving a noun under its respective topic with a random sentiment value.

3.7 Answer Retrieval Structure

The retrieval structure was created to complement the generative model & template system for three reasons. Firstly, retrieval and generative methods complement each other's weaknesses. Retrieval methods provide high quality human written text but are limited by the existing information in the retrieval database. Generative methods can then generate responses outside of the scope of the retrieval structure but suffers from lower quality responses. Secondly, in the work revolving XiaoIce [59], it was found through human evaluation that a hybrid retrieval-generative chatbot performed better than a retrieval and generative systems individually. The average rating of generated responses in the test were 1.09 for the hybrid system, 0.87 for the retrieval system and 0.40 for the generative system. The possible rating for each response was between 0 and 2, where 1 represented an acceptable answer, and 2 represented a good answer. Finally, the fine-tuned GPT-2 model required about 7-8 seconds on average (with the current script and input parameters) to generate text, or in this case, to generate a reply to a question. An issue with the generated answers from the model is that the answers do not always make sense in relation to the question. The goal of creating a retrieval structure that contains answers for common questions, was to ensure that the system could *correctly* answer the most common questions and to reduce the average response time of the system.

Examples of nonsensical generated answers

Q: are you a machine?

A: yes, I'm from the midwest.

Q: what are you called?

A: Um... honest answer is kinda similar.

Q: What is the name of your best friend?

A: hi friend is gonna be nice vanilla bean pancake... so i almost always keep track of which one i keep... i love vanilla... so I know which i like best.

Q: What is your everyday outfit?

A: I've been brushing up on the Cali surf since I was a kid, so I slouched on a chair near a coffee bar watching Netflix.

Two possible alternatives were considered when creating the retrieval structure. The first possible approach is to take the GTKY-preprocessed dataset after it was manually cleaned, and directly use it as the retrieval database. Although it is a possible solution, the dataset was mainly created as training data for machine learning models. Therefore, some questions exist in the dataset mainly to teach models the format of, 'a question followed by an answer', and additional work would be required to further clean the data. The dataset does not specifically take into consideration the most common questions, as found by the analysis of the GTKY dataset. Furthermore, it is important to reduce repetition in the chatbot's answers to increase the perceived engagingness for users who interact with the conversational agent [3]. As such, for each question type, it is desired to have multiple possible answers which would require manually grouping similar questions such that their answers could be shared.

The second alternative is to make use of the common questions found when analyzing the GTKY dataset, and to generate answers with the fine-tuned generative model. The second alternative is more desirable when considering the repetition aspect as it can generate n number of answers for each question.

As a result of analyzing the GTKY dataset, 652 questions had been written down into a txt file, ordered by the top nouns (3.4).

Example of questions ordered by noun

-time-

What do you do in your spare time?

What do you like to do in your free time?

...

-kind / kinds-

What kind of music do you like to listen to?

...

To create a retrieval structure, the questions were first ordered into general topics instead of nouns. The topics were interest/hobbies/activities, music, books, movies, games, student/study/work, person/personal, food, sports and miscellaneous. Questions that did not fit into any other topic were placed into miscellaneous such as "What is your favorite thing to do at the beach?" or "What's your favorite season?". Duplicate questions were possible if the question contained more than one of the most common nouns and would therefore be listed more than once. For example, the question "Do you have any fun weekend plans?" is listed under "fun", "weekend" and "plans". The retrieval questions were not meant to handle follow-up questions, and as such, follow-up questions were discarded as well as duplicates and thereby 343 questions remained.

The remaining questions were iteratively, one by one, input into the fine-tuned GPT-2 model to generate 20 answer samples per question. The results were stored in a csv file, saving each question along with the 20 samples. Out of the 20 samples, some of the samples were duplicates and some answers did not make sense to the question. Therefore, (subjectively) suitable answers were manually extracted. Out of the 343 questions, some additional questions were discarded for various reasons, resulting in 259 retrieval questions. The questions were mainly, but not exclusively, discarded after observing flaws in the associated generated answers. Another reason for discarding questions was due to the question being a type of follow-up question, for example, "What is your dog's name?".

A similar approach to when the template structure was created, was used to assign answer IDs to questions and answers. Questions that have similar meaning were assigned the same answer ID. Thereafter, the respective answers were assigned the same answer ID. Some answers could be answered by a different question type, and were therefore given an additional "optional" ID. The following example illustrates one answer that fits one question, and another answer that fits two questions.

Q1:what do you like to do for fun during the weekend? - ID = 1

Q2:What do you do for fun? - ID = 2

A1:I like to go out to bars on the weekends, but if I'm stressed out then I'll usually head out to eat. ID = 1

A2:I like to go to the movies. ID = 1, Optional ID = 2

3.8 Chatbot Development

The chatbot was developed to be in form of an interactive application in python. Given a user input, the input is first compared to the template questions. This is done by processing a copy of the input to follow the template sentence structure by extracting any existing nouns and comparing with the sentiment memory, as described in (3.5).

Initially, the system was developed to call ConceptNet's API if no matching noun was found in the user's input. Then for every noun in the extracted nouns list, a call was made

to find the IsA relationship of the noun. However, this feature was disabled due to time complexity as it could add an extra two seconds to the agent's response time in some cases. ConceptNet was then replaced by using spaCy's word similarity to compare the similarity between extracted nouns and existing topic words in the sentiment memory. This was done by using the concept of word embeddings, e.g. Word2vec, to create a vector representation of each word. The extracted noun was classified as the topic to which it had the highest cosine similarity to, which allows users to ask what the agent thinks about any noun in any topic. For example, "Q: What do you think about money?". The subject noun was then saved in the sentiment memory with a random sentiment under the topic that it had the highest similarity to, and the agent was able to say whether it liked or disliked the subject.

The processed input is then transformed into an embedding vector, using Universal Sentence Encoder [6]. The template questions are all turned into embedding vectors and cosine similarity is calculated between the processed input and each template question. The question template with the highest similarity is saved along with the cosine similarity value. Then, the original user input is similarly compared to the questions in the retrieval database. Individual threshold values are used for the template and retrieval structure. If both similarity values are lower than their respective threshold, then the question is considered to not have found a similar question in either structures, and therefore an answer is generated by the GPT-2 model. This can be compared to XiaoIce [59] where a pre-set threshold is used to determine candidate responses. A difference is that XiaoIce randomly selects a response out of a pool of responses which have a ranking score above the threshold, compared to selecting the highest ranked response. The threshold values were in this case determined by, during development, testing how low the similarity can be and still return answers that seem relevant enough to be considered acceptable as output. If one or both values are above the threshold, then the question with the highest similarity, returns their respective answer. If the question from the retrieval database has highest similarity, a random answer with the corresponding answer ID is returned. If the template question has higher similarity, then the output needs to be processed further. The output is processed by choosing a random answer template with the same answer ID corresponding with the question. A dataframe data structure in the python library Pandas is used to retrieve information from csv files. The answer when retrieved from its pandas dataframe contains information about how the answer should be processed. The information tells whether the answer should use the same noun as extracted from the user sentence, if it should use the same sentiment word (like/dislike/.../) as the user and how many nouns it should retrieve in a topic (a number of favorite nouns). The instructions are stored in the columns 'use_noun', 'use_sentiment' and 'fetch_count' of the csv file, which is shown in A.5, where integer 0 is mapped to "False" and 1 is mapped to "True".

A user specific memory was developed related to previously answered questions. Whenever a retrieval answer is returned as the reply message of the chatbot, the question, answer and answer ID is saved. If the user asks a question with the same answer ID, then the saved answer is returned, which ensures consistent answers, and a sense of persona.

Example of retrieval memory

Q: What is your name?

A: My name is John.

Q: What is your name?

A: I thought I told you previously. My name is John.

The generative model generates answers whenever the similarity of the template and retrieval questions are too low compared to the user's question. The model is called to generate an output with the user's original text as input together with a `<|startoftext|>` token prepended to the user's text. For example, "Q: `<|startoftext|>` What is the meaning of life?". The model then generates n number of answers, where the first non-empty answer is returned as output.

Internal Testing and Revisions

Once the interactive chatbot was developed, some testing was done internally by letting persons other than the developer interact with the system. It became apparent that the generated answers can sometimes be “faulty” by consisting of `<|startoftext|>` and `<|endoftext|>` tokens. The faulty answers are not formatted as only the answer but may also contain the input question or the start of another question as well as the answer generated by the model. This has to do with how the model is trained on question and answer data. Therefore, this led to the addition of various processing. First, the sentence is split at the first occurrence of “end-oftext” and all the text after the split is discarded. Then any remaining tokens are removed (`<|startoftext|>`, `<|endoftext|>`, `<|`, `|>`). Further processing was performed to ensure that the answer does not contain a question by splitting the text into sentences and taking any sentences before the question. After preprocessing, the first answer that is longer than 1 character (not empty) is returned as the output. It was also observed that the model would sometimes only generate a dot. This would occur when the text given by the user does not contain punctuation at the end of a sentence (question mark, exclamation mark, full stop). The question answering component was developed for question answering and as such can only be expected to work as intended if the user inputs questions. Therefore, a processing rule was made to the user text that adds a question mark to any input that does not end with a form of punctuation. This addition was only added when given as input to the generative model whereas the original user input is saved (recorded) when applicable, such as during user tests.

Additionally, the agent did not have a greeting or a farewell reply, which was then added to the retrieval structure. Finally, an initial message was displayed as the first message of the chatbot which greeted and invited the user to ask the agent about anything. The message was then revised to invite the user to ask anything about the agent and its interests, see figure A.2 in the appendix. The reason for changing the message was an attempt to narrow down the questions the users would ask the agent into the personal question domain that it was designed for.

Artificial response delay

The generative component took around 4-8 seconds to generate a response depending on hardware, number of responses to over-generate, and response lengths. Therefore, artificial response delay was introduced to mask any difference in response time between the generative component and the template and retrieval components, which are near instantaneous (without artificial delay).

Artificial delay has been previously studied in the work of [14]. In the study, it was found that users perceived a chatbot (in the context of customer service) with response delay as more human-like and more socially present compared to the same chatbot with near instantaneous responses. Therefore, the testing objective of whether adding artificial delay in the agent’s response increased the user’s perception that the agent was real (realness), was added to the user test. Testing for this objective was done by randomly using delay for every other user (aiming to be equal numbers of participants in each test group). The hypothesis was that the user group interacting with the agent that has artificial delay, would perceive the agent to be more human-like. The artificial delay was dynamically determined by the length of the agent’s reply, see eq. 3.1.

$$base_delay = number_of_tokens / words_per_second \quad (3.1)$$

The words per second was set at 2 which would represent a very fast typer (120 wpm). The artificial delay only considers how long it would take for a person to type the message and does not take into consideration the time it would take for a person to process the information written by the user, as well as the cognitive process of thinking about how to reply. In contrast, the delay in the work of [14] is calculated based on the natural language complexity of the previous message (from the user) and the complexity of the chatbot’s response.

A threshold for the delay was set at 8 seconds, messages that were calculated to take longer than 8 seconds were suppressed using eq. 3.2. Example delays are illustrated in table 3.2. The threshold of 8 seconds was chosen due to the generative component usually taking roughly 8 seconds to over-generate 20 answers. If the time it takes for the reply to be retrieved or generated is longer than the calculated dynamic delay, then no extra delay was added. Otherwise, the agent waited until the computation time + wait time became equal to the calculated delay time.

$$delay = delay_threshold + \ln(base_delay) - \ln(delay_threshold) \quad (3.2)$$

Words/token	artificial delay
10	5 s
20	8.223 s
40	8.916 s

Table 3.2: Example of artificial delay for different sentence lengths.

3.9 User test - Environment

A front-end for the question answering chatbot was made to make the chatbot more easily available for user testing purposes. The front-end was made using Flask¹¹ together with Ngrok¹², as inspired by a previous thesis at University of Twente [22]. Flask is a web application framework which makes it possible to locally host python applications. Ngrok is a cloud service that can be used to redirect traffic to a local network. The chatbot is therefore made available by locally running a flask application on a specific port and creating a tunnel through Ngrok to the specified port on the local machine. Ngrok will then generate a link with a randomized subdomain, e.g. <https://d332866f.ngrok.io>, where “d332866f” is the randomized subdomain. It is also possible to reserve subdomains by upgrading membership from the free version to a paid version. Ngrok also provides a “local inspector” where it is possible to get an overview of all the HTTP requests, e.g. see questions by the users and the answer by the chatbot.

An initial flask application was created by partially following an online tutorial¹³. By following the tutorial, an HTML page was developed where it is possible to interact with the question answering chatbot. When a user sends a message on the html page, a python function is called that receives the user’s message as input and returns the chatbot’s reply as output. Thereafter, a type of feedback and additional html pages were added. Since the chatbot may take a few seconds if it generates a reply with the GPT-2 model, feedback was designed in the form of “loading dots” to reassure the user that the chatbot is working. After the user sends a message, “animated” dots will be displayed to illustrate that the agent is working on retrieving or generating a reply. Figure 3.3 depicts the mechanic where, once the agent returns a reply, the dots are replaced by the agent’s message. The feedback can be designed in other forms, in this case loading dots was chosen as it is easy to implement and common in some messaging apps, e.g. Messenger.

The additional html pages that were added were:

- A starting page
- A survey page
- An end page

¹¹<https://flask.palletsprojects.com/en/1.1.x/>

¹²<https://ngrok.com/>

¹³<https://dev.to/sahilrajput/build-a-chatbot-using-flask-in-5-minutes-574i>

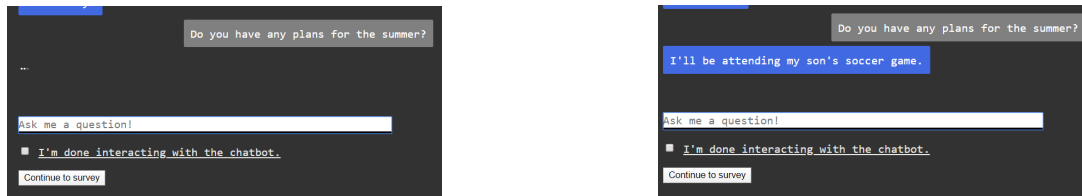


Figure 3.3: Feedback using "loading dots"

- A statistics page

The appendix shows some of these pages (A.6). The starting page informs the user about the user test in terms of their rights and how their data is saved and used in the study as well as instructions regarding how to interact with the agent. The starting page includes a checkbox where the user agrees and gives their consent to the terms of the study. If the user gives their consent, then they are redirected to the chatbot interaction page. The user decides how long to interact with the agent, whereafter they are redirected to the survey page. The survey page consists of the survey questions, which are presented in (3.10.1). Once sending in the form, the user is redirected to the end page, where they are informed that the test is over and that they may leave the page. The statistical page is made for the researchers to have a live overview of the number of unique visits and the number of completed surveys.

Cookies are used to temporarily store user information. When a user enters the front page for the first time, they are given a unique integer user ID which is stored as a cookie, as well as a boolean valued cookie. The boolean valued cookie decides whether the user interacts with an agent that has artificial response delay or not, as described in (3.8). When the user sends messages to the agent, the message history is also temporarily stored as a cookie. Three additional cookies are used, one cookie stores the timestamp of when the user continued to the survey page, while the other two store information regarding the delay in the chatbot's response. One keeps track of the maximum delay that the user experienced, and the other keeps track of the average response delay. The message history cookie is updated after every message sent by the user. The cookie is a python list where each entry contains the user's message, the chatbot's message, the timestamp when the user's message was received and the timestamp just before returning the chatbot's message. The delay cookies are also updated accordingly after each message.

Data is saved on the server side at different steps of the user test. Once the user is given a user ID, the user ID is stored in a csv file together with the current timestamp. When the user is done interacting with the chatbot and redirected to the survey page, the message history is saved in another csv file. The data saved to the csv files is: user ID, the value of the timestamp cookie, the value of the message history cookie and the number of messages sent by the user. At a lower level, the chatbot always saves every answer that is obtained through generating with the GPT-2 model along with the corresponding question, in a pandas dataframe. The chatbot also stores a pandas dataframe related to the retrieval database, such that if a user asks a question which leads to a retrieved answer, the question, answer and answer ID is saved in a user specific memory. The purpose of the memory is to ensure a consistent persona. If the user asks what the agent's name is, then the agent will first randomly retrieve a name and thereafter remember the name and re-use it if the same user asks again. Therefore, the pandas dataframes are saved into their respective csv files at this point as well. The data could be saved once at the end of the user test instead, but this ensures more safety in case the application runs into any unexpected errors. Finally, once the user finishes the survey and submits the data, the survey answers are saved along with additional data into another csv file. The additional data contains the user ID, the timestamp from the related cookie, the average delay and max delay from the delay cookies as well as which version of the test the user interacted with.

When there exist multiple threads that have access to shared resources, it is important to lock the resource when it is used by one of the threads to avoid corrupting data; this is known as mutual exclusion (mutex) in concurrent programming. An example is given below how data may be corrupted.

To update a variable the variable needs to be read from main memory, incremented on local memory and saved back to main memory. If two threads were to update the same variable concurrently without locks, there may be many different inter-leavings of these three steps with different results. For example, with a variable B and two threads called to increment the variable concurrently:

```
Variable B = 5
Thread 1 reads B, B = 5
Thread 1 increments B locally, B = 6
Thread 2 reads B, B = 5
Thread 1 saves B, B = 6
Thread 2 increments B, B = 6
Thread 2 saves B, B = 6
```

The expected output is 7 but the result will be 6 in this case.

In the system, each user interacting with the agent has their own thread and each user (or thread) may read or write data from/to a few csv files. Therefore, mutex locks are used to lock shared resources to only allow one thread to read or write at the same time. This is done by placing a call to acquire the lock before a shared resource and a call to release the lock once the thread is done using the resource. If a thread tries to acquire the lock while it is already acquired by another lock, the thread will pause its execution (blocking) and wait in a queue until the resource is ready.

The generative model was unable to run concurrently. The reason was later found (4.6.7) to be due to the "gpt_2.py" script from the gpt-2-simple library, where the function to generate text re-loads the model, which causes problems when called concurrently. Before the discovery of the cause, a solution was found to load the model multiple times into their own TensorFlow sessions. This was done by duplicating the model on the computer's hard disk and thereafter changing the TensorFlow variables into a different "scope". Then, a queue system was developed by using locks to only allow one user to use one of the models at the same time. Loading multiple models was expensive in terms of VRAM as one model's size was near 500MB. Therefore, 4 copies of the generative model were made available in the queue system for the user test. The choice of 4 copies was sufficient for the scope of the user test where a low number of subjects participated at the same point in time. A script in the library "gpt-2-simple" called gpt_2.py was successively changed to work with the impromptu solution, by adding a "scope" parameter to the 'load' and 'generate' functions, such that the copies of the model could be successfully loaded and used to generate. If more than 4 users, e.g. 5, tried to call the generative model at the same time, then the fifth user would be placed in a queue waiting for a model to become available. This would lead to longer responses times if several users interacted with the chatbot concurrently.

3.10 User test - Question Answering Chatbot

The purpose of the test for the question answering chatbot was to see how users would interact with the chatbot, and to find out how the chatbot was perceived, primarily in terms of realness and engagingness, from the users' perspective. The aim of the user test was to also reveal any potential faults in the system as well as what parts of the system can be further

improved. An additional testing objective was added during development which was about measuring how adding artificial response delay affects the users' perception of the chatbot.

3.10.1 Survey

The survey part of the test instructs users to rate a number of statements on a 5-point Likert scale from 1-“Not at all” to 5-“Very much so”. The statements were:

- The chatbot is engaging/interesting to interact with. (engagingness)
- The time it takes to get an answer negatively affects my chatting experience. (unresponsiveness)
- It feels as if i'm chatting/talking with a real person. (realness)
- The chatbot gave contradicting answers. (inconsistency)
- The chatbot's answers are related/relevant to my questions. (relevancy)
- It feels as if the chatbot's answers are repetitive. (repetitiveness)
- The time it takes to get an answer from the chatbot is acceptable. (responsiveness)
- I would like to interact with the chatbot again in the future. (retention)
- It feels as if the chatbot has its own personal background and personality. (persona)

Some of the listed questions are only relevant for evaluating the question answering chatbot, such as whether the chatbot gave contradicting *answers*. However, to be able to compare the results with other chatbots in the thesis and to draw conclusions of the research, some questions are applicable across all surveys such as questions that measure engagingness and realness.

Moreover, the users were asked to enter their age, their perceived English proficiency and whether they had interacted with a chatbot previously: “Have you previously interacted with a chatbot?” (yes/no).

The options for choosing English proficiency were given in the following order:

- Native speaker
- Fluent, C1 - C2
- Advanced, B2 - C1
- Intermediate, B1 - B2
- Beginner, A1 - A2

The options were transformed into values, 1-5, where 1 represents ‘beginner’ and 5 represents native speaker.

3.10.2 Method

The user test was made available online from 29th of April around 20.00 CET until 11th of May around 10.00 CET. The majority of test subjects participated between the dates 29/4 - 1/5. The user test was first posted in a Facebook group consisting of over 500 students (current students and past students from Linköping University) on the 29th. It was then posted in another student group the following day. On the 5th of May, the test was shared with a small group of researchers that work in related areas. The last subject participated on the 8th, a Friday, and the test was thereafter taken down on the following Monday.

The system ran locally on a laptop where users could access the test through a tunnel by using Ngrok. The laptop had 16gb of RAM, a NVIDIA GeForce RTX 2060 graphics card with 6GB VRAM and an Intel Core i7-9750H CPU @ 2.60GHz CPU with 6 cores.

Instructions

When the link to the user test was posted in the Facebook groups, it was posted with the following message:

“Hello there,

I’m developing a chatbot as part of research for HMI, and I therefore ask you to test my chatbot. I would appreciate it if you could ask at least 5 (personal) questions to my chatbot and thereafter fill in a survey, rating 9 statements on a 0-5 scale. It should take between 5-10 minutes to participate.

The test is expected to be available until the end of the week. (Ps. may not be suitable for mobile devices) “

The users were instructed to ask at least 5 questions to try to get to know the agent, as if it was a real person. They were also told to send complete sentences, one at a time. Once they decided that they were done asking the agent questions, the users were then finally instructed to fill in a short 10 question survey.

3.10.3 Hypothesis

Previous work has observed positive effects on users of using dynamic response delay in customer service chatbots, compared to instantaneous responses [14]. Therefore, two hypotheses were formed:

1. Users perceive chatbots with artificial response delay to be more human-like (realness), compared to the same chatbots with instantaneous responses.
2. Users perceive chatbots with artificial response delay to be more engaging to interact with, compared to the same chatbots with instantaneous responses.

3.10.4 Results

The user test had 32 participants who interacted with the chatbot and successfully filled in the survey. The participants are assumed to be mostly students and some researchers, as a link to the test was mainly shared in student related groups on Facebook. When asked “Have you previously interacted with a chatbot?”, 28 participants answered yes. The participants were asked to estimate their English proficiency from beginner to fluent, or if they were native speakers. The majority of participants estimated themselves to be fluent speakers and the average English proficiency was calculated as 3.875 (SD=0.609). The average age was calculated as 24.344 (SD=3.395). The user test was divided into two groups, DELAY and NO-DELAY with 16 participants in each group. A participant was assigned to one of the groups based on if their unique integer ID was odd or even.

The number of messages sent to the agent was 240 in total with an average of 7.5 messages per interaction. Out of the 240 messages, 199 were generated, 15 were template answers and 26 were retrieved answers. 11 of the 32 subjects purely interacted with the generative model.

As for the system’s response time, the time for the agent to send out an output after receiving an input, was calculated as 5.983 seconds on average (all messages considered). The average response time for the DELAY group was calculated as 6.646 seconds compared to 5.103 seconds for the NO-DELAY group.

One of the subjects in the NO-DELAY group interacted with the agent for over 100 messages and is therefore considered an outlier. This subject is not included in the 240 messages mentioned previously or the response time calculations.

After analyzing the message history between the participants and the agent, only 16 (50%) of the participants interacted with the agent according to the instructions. By not following the instructions, the participants would interact with the agent outside of its designated application. Some individuals requested the chatbot to ask them a question. Other individuals asked non-personal questions, and others would merely interact with the agent to try to understand how the agent worked.

The following questions or rules are used as guidelines to divide the samples into “ideal”, “border-line” and “out-of-scope” groups. For the most ideal case, the answer to the first question is ‘yes’ and then ‘no’ for the second question.

1. Were the majority of the questions asked, personal questions?
2. Did the user data contain non question messages?

The first rule is mainly considered whereas the second rule is less enforced to allow some reactionary messages, such as laughter or acknowledgement. The “border-line” group consist of interactions where the user may partially ask some personal questions but also task-oriented questions or interacting with the agent to explore its capabilities. The “out-of-scope” group consists of interactions where users clearly broke the two previously mentioned rules, e.g. by not asking enough personal questions and sending multiple non-question messages. Users that did not follow the instructions are not considered a valid representation of the question-asking component for its intended application. As such, the “border-line” and “out-of-scope” groups are merged into a “non-ideal” group. The border-line group consisted of 8 participants and the out-of-scope group consisted of 8 participants as well. Examples of conversations from the three groups (ideal, border-line, out-of-scope) are provided in the appendix (A.7).

Table 3.3 shows the survey results from different groups, with the number of participants in each group shown in parenthesis in the first row. T-tests were performed to find any significant differences in mean values between groups. The resulting p-values are shown in table 3.4. With a significance level of $\alpha = 0.05$, it was found that there was a significant difference between the ideal and non-ideal groups on a majority of the measures (responsiveness, realism, relevancy, unresponsiveness and persona). It was hypothesized that the DELAY group would find the agent to be more engaging and human-like than the NO-DELAY group. Instead, the opposite was observed with statistical significance ($p=0.037$) that the ideal NO-DELAY group perceived the agent to be more human-like than the ideal DELAY group. As for engagingness, the null hypothesis could not be rejected as there was no significant difference between the groups. Furthermore, it was found that there were significant differences between the ideal DELAY group and ideal NO-DELAY groups on the measures of inconsistency and relevancy.

Due to the unexpected findings, further investigation was performed by analyzing the conversations in the ideal group. It was found that roughly 59.6% of the agent’s responses made sense with respect to the user’s input in the ideal DELAY group compared to 64.4% percent in the ideal NO-DELAY group. The ideal DELAY group consisted of 7 subjects with 6.71 inputs on average per conversation, compared to 9 subjects in the ideal NO-DELAY group with 6.56 inputs on average. The ideal DELAY group asked (roughly) 15 follow-up questions in total, where each user asked at least one follow-up question. The ideal NO-DELAY group asked (roughly) 13 follow-up questions in total, with 3 out of the 9 subjects not asking a single follow-up question. The ideal DELAY group had 3 subjects who interacted with the template or retrieval components, with 8 responses in total from those components. In the ideal NO-DELAY group, every subject got at least one response from the template or retrieval components, adding up to 23 total responses from those components. The results are discussed in chapter 8.

Group	All (32)	Ideal (16)	Non-ideal (16)	All DELAY (16)	Ideal DELAY (7)	All NO-DELAY (16)	Ideal NO-DELAY (9)
engaging-ness	3.281, (1.301)	3.688, (1.138)	2.875, (1.360)	3.188, (1.109)	3.143, (1.215)	3.375, (1.500)	4.111 , (0.928)
unresponsiveness	2.250, (1.391)	1.625, (0.885)	2.875, (1.544)	2.062, (1.436)	1.286 , (0.488)	2.438, (1.365)	1.889, (1.054)
realness	1.969, (1.177)	2.375, (1.147)	1.562, (1.094)	1.625, (1.025)	1.714, (0.951)	2.312, (1.250)	2.889 , (1.054)
inconsistency	3.188, (1.281)	2.750, (1.183)	3.625, (1.258)	3.438, (1.209)	3.429, (1.134)	2.938, (1.340)	2.222 , (0.972)
relevancy	2.750, (1.136)	3.250, (1.125)	2.250, (0.931)	2.312, (0.793)	2.429, (0.787)	3.188, (1.276)	3.889 , (0.928)
repetitiveness	1.938, (0.982)	1.750, (0.931)	2.125, (1.025)	1.812, (0.750)	1.571 , (0.787)	2.062, (1.181)	1.889, (1.054)
responsiveness	3.812, (1.355)	4.312, (1.014)	3.312, (1.493)	3.812, (1.276)	4.143, (1.069)	3.812, (1.471)	4.444 , (1.014)
retention	2.938, (1.458)	3.188, (1.515)	2.688, (1.401)	2.688, (1.493)	2.571, (1.718)	3.188, (1.424)	3.667 , (1.225)
persona	2.750, (1.391)	3.250, (1.528)	2.250, (1.065)	2.438, (1.263)	2.429, (1.512)	3.062, (1.482)	3.889 , (1.269)

Table 3.3: Survey result of the first user test, illustrating the mean and standard deviation for different groups.

Group	engaging-ness	unresponsive-ness	realness	inconsistency	relevancy	repetitive-ness	responsive-ness	retention	persona
Ideal DELAY	0.092	0.185	0.037	0.038	0.005	0.518	0.573	0.158	0.054
All DELAY	0.690	0.455	0.099	0.277	0.027	0.480	1.000	0.340	0.209
Ideal - Non Ideal	0.077	0.009	0.049	0.052	0.010	0.287	0.034	0.340	0.040

Table 3.4: P-values when comparing survey results of a) Ideal DELAY group against Ideal NO-DELAY, b) All DELAY group against All NO-DELAY group, c) Ideal group against Non-ideal group.

4 | Development - Post User Test

After analyzing the user test, it was observed that the vast majority of replies, over 82% (199/240), returned by the agent was from the generative model. It was also observed that users oftentimes asked follow-up questions (A.7). Therefore, a decision was made to focus on improving the generative model by:

- Developing a ranking system for over-generated answers, since some generated answers are more or less relevant to the question than others.
- Training a new model that takes history into consideration, in hopes of better being able to answer follow up questions by having short-term memory and context understanding.

Training a new model consists of creating new datasets, training different models on the new datasets and running tests to compare the models against each other and against the existing model (model selection).

Other than improving the generative model, time was spent on refactoring the system, updating the template component and creating question and answer classifiers.

4.1 Refactoring

Before working on improving the generative model, some refactoring of code took place. The agent consists of three components: templates, a retrieval database and the generative model. The components were split into their individual python files instead of residing in the same python file. Another file was created to contain common models, functions and data that was shared between the components. By isolating the components, the aim was to make the system more understandable and to allow for ease of component-wise development and testing. If one component had bugs, then the component could run by itself to find out why the bug occurred.

4.2 Template Component Improvements

From the user test, it was observed that one type of question intended for the template and sentiment component had not been taken into consideration during its design. The question observed was “What is your least favorite color?”, to which the question was not recognized as a template question, and therefore the generative model generated an answer. This resulted in two changes. The first change was to add two more template questions with their respective answers.

New question templates:

- What is your least favorite <topic>?
- Do you have a least favorite <topic>?

New answer templates:

- My least favorite <noun> is <noun_1>.
- I do, my least favorite <noun> is <noun_1>.

The second change was to implement sentiment analysis to determine whether the user asked if the agent likes (positive sentiment) or dislikes (negative sentiment) something. The initial naive method first developed made an assumption, based on observations from the conversations in the GTKY dataset, that the user is asking what the agent likes. Then, the user's input sentence is tokenized and each token is compared to a list of sentiment keywords (like, love, dislike, hate), to naively find if there is a positive or negative sentiment. The updated method instead implements VADER sentiment analysis. VADER is developed for short text sentiment analysis [19] and was observed to work well for this specific use-case. Two other sentiment analysis tools were tested: Stanford sentiment¹ and TextBlob². Neither of the two were able to give non-zero sentiment to short texts, such as "Do you like apples?" or "I like apples.". Previous research has compared different sentiment analysis methods and VADER was one of the best methods on social media related datasets [34].

Example of VADER sentiment analysis:

- What is your least favorite color? Sentiment: NEGATIVE (-0.357)
- Don't you like apples? Sentiment: NEGATIVE (-0.2755)
- Do you like apples? Sentiment: POSITIVE (0.3612)
- I neither like them nor dislike them. Sentiment: NEUTRAL (0.0191)
- I like sports. Sentiment: POSITIVE (0.3612)
- What food don't you like? Sentiment: NEGATIVE (-0.2755)

Another change was done while refactoring the code for the template structure which was to enable the occurrence of similar topic words for finding the topic. Previously, a topic was found by matching the exact word with the topic word from the sentiment memory. The change was to instead use spaCy's token similarity measure. By briefly experimenting during the development, a threshold for the similarity could be determined as 0.65. Words above the similarity threshold were translated as the topic to which the word had highest similarity. Thereafter, 'colour' could be translated as the topic 'color', 'film' could be translated into the topic 'movie' and 'series' could be translated to the topic 'TV show'.

4.3 Follow-Up Question Test

A short test was performed on the current model (#10 in table 3.1) to find if it was able to take in one previous question and answer pair in addition to the new question, as input when generating an answer. This test was very briefly performed to find if the model had context understanding without having been fine-tuned for it. It was observed that the model could in fact occasionally generate answers to follow up questions. It can be observed that most answers are contradictory or makes little sense whereas some answers could work as an answer. This also illustrates the need for a ranking of answers.

Example 1

Generative model input:

<|startoftext|>What do you think about abortion? Hm...I don't really like abortion that much. <|endoftext|> <|startoftext|>why not?

Output (possible responses):

¹<https://nlp.stanford.edu/sentiment/>

²<https://textblob.readthedocs.io/en/dev/>

- Because it might be safer for the baby.
- because it might help with your ADD and other stuff I guess? It might help with your immune response to certain types of viral infection.
- Because it may be cheaper to have an ectopic pregnancy.
- because it might end up with the baby they're planning on having, not the mother.
- Because it might be cheaper and easier to get an abortion in NY.
- It would be nice if there were more clinics by the time the local one closed.
- It might be easier to handle if you know what's best for you.
- Because it can be very difficult to navigate through the abortion industry.

Example 2

Generative model input:

<|startoftext|>What do you think about soccer? I'm not very fond of it. <|end-oftext|> <|startoftext|>why not?

Output (possible responses):

- because it's fun to watch and to relax.
- I don't really like to play soccer.
- It's difficult to tell without looking at the scores.
- because u are a guy.
- It's difficult to tell... it's pretty simple: ballgames, stats, practice, games, stats.
- because I don't like the ball hockey has to offer.
- Because it's too easy a game to miss while still having enough time for the teams play.
- Because it's very slow-paced and involves lots of movement around corners.

4.4 Answer Ranking

It has been observed throughout the development that the generative model generates some answers that have little to no relevance and some answers that are relevant, when given the same input. Therefore, it was considered necessary to have a ranking. A ranking was developed where answers could be over-generated by the generative model and then ranked to find the most appropriate and/or interesting answer to the question.

A related work [54] which developed a hybrid retrieval-generation neural conversation model made use of BM25 for ranking candidate responses from the retrieval component. The system then used a neural network for re-ranking responses to return the highest ranked response as the output. Inspired by the previous work, BM25 and Neural-Network-based rankers were investigated, as well as another method called Latent Dirichlet Allocation (LDA). After the investigations, a decision was made to develop a novel ranker based on cosine similarity with additional penalty and reward functions, as described in section (4.4.4).

4.4.1 BM25

Initially BM25, which is a "best-match" ranker used by search engines [35], was investigated whether it was able to be used to rank the over-generated answers. A python library called rank-bm25 was used as it consists of a few different versions of BM25 such as Okapi BM25. We consider the generated answer sentences to be the corpus for the algorithm, and the user's question to be the query. The algorithm ranks the documents, or the answers in the corpus

based on words in the query (question). If no matching words are found then every answer gets a score of 0, which would result in arbitrarily picking an answer. To rank answers solely based on matching words is not desired as it cannot recognize answers that may be relevant, but that lacks shared words with the question. Therefore, other options were considered.

4.4.2 Neural Network Ranking

While reading about which rankers are used for ranking answers, it became apparent that some researchers worked with creating rankers with the help of neural networks, such as aNMM [53]. Then, a toolkit called MatchZoo³ was found where tools for different types of classification and ranking in natural language processing contexts exist. The toolkit had a way to train and use various neural network rankers, such as aNMM.

Training data was constructed by using the retrieval database. Each question from the database was matched by a number of correct and incorrect questions, see figure 4.1. The correct answers were labeled 1 and were found by matching the answer ID related to the question. Incorrect answers were taken by randomly fetching an answer from the database and labeling the answer 0 if the answer ID did not match the question. Then, by following the steps provided by the toolkit's tutorial⁴ different models were trained, such as the aNMM as well as the toolkit standard "DenseBaseline". However, the maximum achieved prediction accuracy (correctly labeling an answer to the question) was only near 68%. This might have been explained by the low amount of training data in relation to the number of trainable parameters in the models. However, after reducing the trainable parameters from 319,862 down to 2,162, for the case of the aNMM model, no significant change was observed. The parameters in the 'DenseBaseline' model was reduced gradually from 163,713 trainable parameters down to 344, yet no significant change was observed. The idea of using a neural network ranker was therefore put on hold as the trained models were deemed unreliable.

	id_left	text_left	id_right	text_right	label
0	Q0	what do you like to do for fun during the week...	A0-0-0	I like to go running/hiking.	1
1	Q0	what do you like to do for fun during the week...	A0-0-1	I work with web development.	0
2	Q0	what do you like to do for fun during the week...	A0-1-0	I like to play tennis during the day but I als...	1
3	Q0	what do you like to do for fun during the week...	A0-1-1	I'm not sure.	0
4	Q0	what do you like to do for fun during the week...	A0-2-0	I usually go to the beach.	1
...
5597	Q274	where are you from?	A274-11-1	My hobbies include biking and reading.	0
5598	Q274	where are you from?	A274-12-0	London, the UK.	1
5599	Q274	where are you from?	A274-12-1	I'm doing just fine.	0
5600	Q275	How are you?	A275-0-0	I'm doing well.	1
5601	Q275	How are you?	A275-0-1	nothing too much planned.	0

Figure 4.1: Training data for a neural network ranker

4.4.3 LDA

Latent Dirichlet Allocation [4] (LDA) is a type of topic modeling. It assumes that a set of words make up a topic, and that each document is a mixture of topics. It was found that LDA had been previously used for finding topic relevance of over-generated questions [7], which is not too different from over-generating answers.

³<https://github.com/NTMC-Community/MatchZoo>

⁴https://github.com/NTMC-Community/MatchZoo/blob/master/tutorials/quick_start.ipynb

As such, the idea was to use LDA to classify the topics of over-generated answers and the user's question. Then, rank the answers based on topic distribution similarity to the question. A vector could be created for the question and each answer, and then cosine similarity could be used. Each value of the vector would consist of the percentage of how much a document (text) belongs to a topic. For example, with three topics and a document with the topic distribution where 60% belongs to topic 1, 40% belongs to topic 2 and 0% belongs to topic 3 then the vector would be (0.6,0.4,0.0).

A brief test was performed to see whether LDA could be used to find distinct topics in the GTKY dataset. A tutorial⁵ was followed that made use of the LDA implementation from a python library called gensim. However, the LDA option ended up not being fully explored, as the ranking aspect of it was abruptly replaced by a new idea which revolved around using the Universal Sentence Encoder [6] to create sentence embeddings and calculating cosine similarity. The topic classification aspect was replaced by clustering sentence embeddings and the creation of classifiers by using the cluster labels. One reason that the LDA option was not fully explored was that it was unable to create a larger number of distinct topics, see figure 4.2. The topics were observed to have some overlap and were less distinct than previously observed in the clusters created during the data analysis part (3.4). The words that make up each topic, with their related weights, are illustrated in figure 4.3. It also became apparent that LDA works better for longer texts as longer texts may contain more topics, whereas shorter texts, such as tweets or in this case sentences, mainly consist of one topic⁶. However, altered LDA algorithms [55] have been made specifically for short text topic modeling which may be an option to consider, if LDA was to be used.



Figure 4.2: LDA topic distance when using 10 topics on the GTKY dataset.

4.4.4 Cosine Similarity With Penalty and Reward Functions

After trying BM25, neural network rankers and LDA, a novel ranking system was created based on the cosine similarity between (generated) answers and the current question by the user. The similarity ranker made use of a word length penalty function, a repeated key-

⁵<https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/>

⁶<https://towardsdatascience.com/short-text-topic-modeling-70e50a57c883>

```
[
  (0,
    '0.123*plan" + 0.056*weekend" + 0.047*kind" + 0.047*pet" + 0.040*place" '
    '+ 0.037*nice" + 0.033*dog" + 0.025*always" + 0.024*visit" + '
    '0.023*world)'),
  (1,
    '0.089*fun" + 0.072*snow" + 0.052*use" + 0.048*see" + 0.046*know" + '
    '0.041*still" + 0.032*bad" + 0.028*better" + 0.027*hot" + 0.027*new)'),
  (2,
    '0.248*today" + 0.116*make" + 0.107*school" + 0.034*prefer" + '
    '0.023*artist" + 0.017*currently" + 0.017*art" + 0.016*side" + '
    '0.016*planning" + 0.010*research)'),
  (3,
    '0.201*go" + 0.078*think" + 0.068*kid" + 0.042*travel" + 0.035*lot" + '
    '0.031*play" + 0.031*often" + 0.027*state" + 0.026*ask" + '
    '0.026*question)'),
  (4,
    '0.103*weather" + 0.088*ever" + 0.063*good" + 0.051*enjoy" + '
    '0.048*thing" + 0.045*type" + 0.044*right" + 0.027*sport" + 0.027*well" '
    '+ 0.023*food)'),
  (5,
    '0.089*job" + 0.069*study" + 0.061*part" + 0.059*much" + 0.050*country" '
    '+ 0.040*money" + 0.039*forum" + 0.039*far" + 0.034*find" + '
    '0.027*survey)'),
  (6,
    '0.315*live" + 0.099*would" + 0.036*area" + 0.032*could" + '
    '0.024*particular" + 0.019*somewhere" + 0.016*regular" + 0.016*choose" + '
    '0.015*cool" + 0.014*speak)'),
  (7,
    '0.116*favorite" + 0.088*get" + 0.057*hit" + 0.051*try" + 0.035*watch" '
    '+ 0.034*many" + 0.033*usually" + 0.033*show" + 0.028*start" + '
    '0.026*never)'),
  (8,
    '0.125*long" + 0.111*time" + 0.070*old" + 0.046*married" + 0.045*child" '
    '+ 0.044*music" + 0.040*family" + 0.036*student" + 0.030*free" + '
    '0.028*morning)'),
  (9,
    '0.196*work" + 0.119*day" + 0.092*hobby" + 0.068*mturk" + 0.059*name" + '
    '0.034*s" + 0.028*want" + 0.027*full" + 0.022*say" + 0.022*interest')]
```

Figure 4.3: Words and weights for 10 topics as determined with LDA.

word penalty function, a rare word reward function and a repeated answer removal function. The similarity measure worked by transforming the user's input (question) and the over-generated answers into sentence embeddings (vectors) by using the Universal Sentence Encoder [6]. Thereafter, the cosine similarity between each individual answer and the question could be calculated. The concept was extended to compare the similarity of answers to not only the current question but also the conversation history. A history embedding was created as a vector that is continuously updated with every new question and answer sentence embedding. The ranking is then instead based on the cosine similarity between the answers and the history embedding. The update is a linear interpolation controlled by a parameter α , as seen in equation 4.1

$$\text{history_embedding} = \text{history_embedding} * \alpha + \text{new_embedding} * (1 - \alpha) \quad (4.1)$$

A small α results in the history embedding being "near-sighted" by mostly considering new input. A larger α results in the history embedding being "far-sighted" by mostly considering the past history. By updating a history embedding in this manner, the hypothesis is that the system will have a sense of the topic that is discussed. The aim was to enable the agent to answer (at least) one follow-up question, especially unspecific ones such as "Why?" questions.

Follow-up question example

Q: Do you like to watch movies?

A: I love watching movies but I don't have a lot of free time as of late.

Q: Why?

The α was set to 0.2, the low α is to ensure that the agent considers the new question the most while still retaining some sense of the topic, when ranking the answers. By running experiments, it was observed that the ranking lead to relevant answers being returned more often when using the ranking compared to when not using any ranking (4.4.5). However, an

apparent downside of using the similarity ranking is that it prioritizes answers with the same word(s) as the user’s questions. This means that answers that are relevant and maybe more interesting are ranked lower than the less interesting answers that make use of the word(s).

Similarity ranking example, similarity score in parentheses

Q: Do you have any pets?

A-1: Yes, I have a pet. (0.594)

A-2: Yes, I have two collies. (0.496)

A-1 is ranked higher as it shares a keyword with the question, but A-2 is considered more interesting and intelligent as it does not use the keyword directly, but is still related to, and answers the question. In this thesis, answers are deemed more interesting if they are longer (but not too long, e.g. longer than two sentences) and avoid sharing keywords when possible. The observed downside of this ranking led to the development of penalty and reward functions on top of the similarity measure as a means of achieving more interesting answers. The following penalty and reward functions were developed to be used together during ranking. Extensive testing may be required to optimize the penalty and reward functions, as their purpose is to enable answers that have slightly smaller similarity scores than the largest one, to be able to surpass the largest one on the ranking. To optimize the penalties and reward(s), it may be necessary to run tests by letting the generative model generate answers for several questions, observe the answers and their respective similarity scores, and then decide what an acceptable similarity range is. By knowing the acceptable similarity range (the border between relevant and irrelevant answers), it is possible to determine how much an answer can be affected by the penalty and reward functions. If this range is not found or known, irrelevant answers risk being ranked higher than relevant ones, which is undesirable.

The first penalty function that was developed was based on the sentence length. The function is given an ideal length of words and any answer with less than or more than the ideal number of words is penalized, see eq. 4.2 and 4.3. In this case the ideal number of words was set to 20, although a dynamic length to match different users’ message lengths is recommended to be further investigated, as inspired by the findings that humans tend to match their response length to that of their conversational partner [36].

$$token_difference = |ideal_tokens - n_tokens| \quad (4.2)$$

$$length_penalty = \begin{cases} \ln(token_difference), & \text{if } token_difference > 0 \\ 0, & \text{otherwise} \end{cases} \quad (4.3)$$

Using the length penalty ensures that, when there exist multiple relevant answers with similar similarity scores, the sentence closer to the ideal length is more likely to be chosen. This function’s main purpose is to reduce the occurrence of answers that are very relevant to the question but are also very short and therefore deemed less engaging.

Ranking example before length penalty

Q: What music do you like?

A-1: I like all kinds of music. (0.749)

A-2: I like all kinds of music but I mostly listen to rap. (0.732)

A-3: I’m not sure but I think I would have to say that Van Gogh is my favorite artist. (0.350)

In this example, by turning the question and answers into sentence embeddings and calculating the cosine similarity of each answer to the question, the similarities for A-1, A-2 and A-3 are calculated as 0.749, 0.732 and 0.350, respectively. By purely using cosine similarity as a ranking, A-1 would be chosen while A-2 contains more information that may make the

conversation more engaging by allowing a follow up question such as “Who is your favorite rapper?”. Therefore, a length penalty is applied. Two options are considered for applying the penalty. One is to first normalize the similarity scores, such that A-1 becomes 1.0 and A-3 becomes 0.0. This creates further “distance” between higher and lower scored answers. The other option, which is used in this case, is to use the similarity score as it is, which increases the risk that smaller similarity scores, e.g. A-3 may surpass A-1 or A-2 with the help of the penalty and reward functions. Regardless of the option chosen, the penalty scores are normalized and a percentage of the normalized penalty score is subtracted from the similarity score. The “percentage” subtraction is another parameter β , in this case β is 0.1. By following the second option, the answer that would be chosen after applying the length penalty is now A-2.

Updated scores:

- A-1: 0.749 -> 0.649
- A-2: 0.732 -> 0.652
- A-3: 0.350 -> (unchanged)

The second penalty function was developed to penalize answers that contain keywords from the question. The penalty intensity correlates to the number of unique shared keywords, see eq. 4.4, and 4.5. This type of penalty may lead to the occurrence of more interesting answers. Using a variation of the previous ‘pet’ example, A-2 would have a higher chance of being the answer by using this penalty function. Although A-2 got a higher chance of being selected, it would not be selected when *only* using the penalty function, but it might be selected when used together with other penalty and reward functions.

Keyword penalty example 1

Q: Do you have any *pets*?

A-1: Yes, I have a *pet*. (0.594) -> (0.527)

A-2: Yes, I have two collies. (0.496) -> (unchanged)

A-3: No *pets*. (0.685) -> (0.564)

However, in many cases, re-using keywords from the question in the answer occurs naturally, and as a such these answers are also inadvertently penalized.

Keyword penalty example 2

Q: Who is your *favorite rapper*?

A-1: I don’t have a *favorite rapper*. (0.800) -> (0.678)

A-2: My *favorite rapper* is Eminem. (0.793) -> (0.707)

A-3: Eminem. (0.645) -> (unchanged)

The function first tokenize the question and all the over-generated answers. Then stop-words are removed by using NLTK’s stop-word dictionary. For each answer, loop through the remaining tokens/words and compare with the remaining tokens in the question. For every unique token in the answer that exists in the question, a counter “shared_tokens” is incremented. A percentage of the calculated penalty is then subtracted from the similarity score. The “percentage” is another parameter ω .

$$\text{penalty_magnitude} = (\text{shared_tokens} / \text{num_q_tokens}) * (\text{shared_tokens} / \text{num_ans_tokens}) \quad (4.4)$$

$$\text{shared_word_penalty} = \ln(\text{const} + \text{val}) - \ln(\text{const}), \text{const} = 2 \quad (4.5)$$

A reward function was developed that rewards answers with less common words as an additional way to counteract answers with the same keyword as the questions, being prioritized. For example, the name of a music genre will be rarer than the word “music”.

Example without reward function

Q: What music do you like?

A-1: I like all kinds of music. (0.749)

A-2: I like to listen to a lot of rap. (0.642)

The reward function was created by using the Persona-chat and the GTKY datasets to create weights for each word by calculating their idf-weight (inverse document frequency). Relating to the previous example, the calculated idf-weight for music is 7.165 while it is 9.733 for rap. The weights were created by first tokenizing the merged datasets and then lemmatizing each token by using spaCy. Two functions from the sklearn library were used; CountVectorizer to create a vocabulary of the data and TfidfTransformer to calculate the idf weights. Finally, a pandas DataFrame was created containing the words and their respective weights, which was then turned into a csv file. The reward function then looks up the weights for each word in an answer, sums up the weights, divides the sum first by the number of tokens and then by the mean idf weight. A percentage of the calculated reward is added to the similarity score, based on a parameter θ .

4.4.5 Ranking Tests

A test was performed by one internal person (a student with a computer science background) to compare answers chosen by three methods:

1. Using no ranking
2. Only similarity ranking
3. Similarity ranking with length penalty.

The test was made by feeding the generative model questions from the user test data and then storing the best answers by the two ranking methods and the top answer for the answer without any ranking. The evaluation was done manually by looking at the question and the answers from each method. The evaluation was performed in a one-on-one style where a point was awarded to the answer that was most relevant to the question. If both answers were equally relevant or irrelevant, then no score was awarded (tie). The distributed points of method 1 against method 2 is shown in table 4.1 while method 2 against method 3 is shown in table 4.2.

Method	No ranking	Ranking	Null (Tie)
Relevancy	1	10	44
Engaging	10	16	29

Table 4.1: Points awarded to Method 1 and Method 2 in a one-on-one style evaluation on relevancy and engagingness of generated answers to 55 questions.

Method	Ranking	Ranking + Penalty	Null (Tie)
Relevancy	-	-	-
Engaging	5	8	42

Table 4.2: Points awarded to Method 2 and Method 3 in a one-on-one style evaluation on relevancy and engagingness of generated answers to 55 questions.

When comparing method 1 and method 2 on 55 questions, it was deemed that method 2 had relevant answers more often than method 1. For 44 out of the questions, no points were awarded. method 1 was awarded one point, whereas method 2 was awarded 10 points. Note that some answers could not be compared due to the question being a follow up question

specific to the original answer from the user test. In such cases, the question was considered a tie. Method 3 was not compared as it is very similar to method 2 in terms of relevancy.

Another evaluation was done to measure which ranking method had the most interesting answers. Interesting answers are in this case considered as answers that contain more substance/information/detail, for example "I like all types of music but my favorite is rap music." instead of "I like all types of music." Although the evaluation may have been subjective, it was done for the purpose of making a fast development choice. With more resources the evaluation could have been performed through utilizing crowdworkers to award points, whereafter an average could be calculated. The score resulted in 10 points for method 1 and 16 points for method 2. When comparing method 2 and 3, method 2 got 5 points and method 3 got 8 points. The lower total of points in the comparison of method 2 and 3 was due to a larger number of tied answers. As the two methods are both based on a similarity measure, a lot of answers were exactly the same.

A conclusion was made that adding similarity ranking ensures that answers are more often relevant to the question than if no ranking is used. Additionally, by using length penalty, more interesting answers were found (slightly) more often.

4.5 Question and Answer Classifiers

Previously during data analysis, clustering had been used to find the most distinct topics of the GTKY dataset. An idea was to make use of the clustering to implement a way to classify what topic a question was. Question and answer classifiers could serve useful for ranking answers, e.g. by only considering answers classified as the same topic as the question. Another potential use-case was to create rules for how the chatbot should respond based on different message types (e.g. acknowledgement or laughter, etc.). The question classifier was developed by first extracting each sentence that ended with a question mark and then turning the sentences into sentence embeddings by using the Universal Sentence Encoder. Thereafter, a function called AgglomerativeClustering from the scikit-learn library, was used to create 20 clusters. Each cluster was manually checked by printing the sentences that belong to a specific cluster. Thereafter the clusters were manually named descriptively based on its content as a complement to the automatic integer labels (0-19), as shown in table 4.3. Some clusters were similar but contained different questions, such as 5 and 11, 6 and 16, as well as 3 and 9.

A classifier could be trained by using the cluster (integer valued) labels as output and the sentence embeddings as input. The data was split into 80% training data and 20% testing data. A hypothesis was made that a distance-based classifier would achieve higher accuracy compared to other common classification methods. The hypothesis was grounded in the fact that sentence embeddings were used, meaning that similar sentences are closer together in vector space. Therefore, a K-nearest neighbor (KNN) classifier was initially trained on the training data. Additional classifiers such as 'decision tree' and 'random forest' were also trained. The KNN classifier achieved the best prediction accuracy on the test data, near 90%, and was therefore chosen. Since some clusters were similar, similar clusters were merged one by one and then a comparison was made to see how the prediction accuracy was affected. This resulted in clusters 5 and 11 being merged as well as 6 and 16. No improvement was observed when merging cluster 3 and 9 and therefore they remained separated. Once the testing was over, a new KNN classifier model was trained on all the data (merged training and testing data).

A brief test was held to find if the classifier could classify answers into the question topics as well. It was observed that the classifier had a more difficult time classifying answers, and therefore a separate answer classifier was developed. Each sentence that did not end with a question mark was extracted from the GTKY dataset and the same method was applied. After examining the 'answer' clusters and the distribution of samples in each cluster, some clusters had a larger number of samples than others, therefore, more clusters were created. A cluster

Cluster ID	Descriptive label
0	hobbies/interests
1	where are you from?
2	kids/married/pets/male/female
3	follow up question type
4	job
5	and you? (reflect question)
6	student/studies
7	movies/series/reading
8	sports/games
9	follow up question type
10	how long...?
11	reflect question
12	food/drink
13	weather
14	how are you?
15	what's your name?
16	study
17	music/instrument
18	really? (surprised reaction)
19	traveling

Table 4.3: 20 (question) clusters formed from the GTKY dataset using agglomerative clustering.

of 40 was instead chosen in this case. Some clusters were removed (1, 5, 8, 11, 14, 15, 21, 23, 38) due to either consisting of a larger range of mixed topics or nonsensical data. The remaining clusters are presented in table 4.4. Cluster 36 consisted of “Nice to meet you” sentences which was removed as it was deemed out of scope for the question answering chatbot. Cluster 17 consisted of sentences such as “I will try that” and was also removed for the same reason as cluster 36. Cluster 28 was removed as it consisted of only smiley faces ‘=)’. An idea occurred that some answer clusters could be used to extend the question classifier. The question classifier was trained once again with 11 more clusters from the answer clustering (36, 34, 33, 30, 29, 27, 24, 22, 20, 10, 7). The new clusters offered classification for various acknowledgements and other common messages such as “Hello”, “Thank you”, “good morning”, etc. Thereafter the answer classification was trained, and the two trained classification models were saved.

4.6 Generative Component Improvements

Improvements to the generative component consists of developing an answer ranking, creating new datasets with conversation history taken into consideration, fine-tuning new models, model selection, and general testing.

4.6.1 Preprocessing and Creating New Datasets II

For the sake of being able to fine-tune a model to have short term memory or context understanding, it is necessary to create new datasets. The main idea was to preprocess the Persona-Chat dataset to retain a history of past questions and answers in a conversation. The Persona-Chat data was first preprocessed into questions and answers, separated by each conversation. Thereafter, while looping through the questions and answers in the conversation, for each future question, the past questions and answers were included in a history section. This was structured by introducing new tokens called “<|startofhistory|>” and “<|endofhistory|>”.

Cluster ID	Descriptive label
0	muscle/car/biking
2	location/residence
3	likes/plans/outdoor
4	reading/studying
6	movies/series
7	good morning/ i'm good
9	travel/language
10	LOL, hahaha
12	work
13	years/numbers
16	dance/music
18	hobbies/interests
19	study/Work
20	"awesome" "cool" "excellent" "that's great"
22	"understandable" "fair enough" "interesting"
24	yes/no/indeed
25	pets
26	food/cooking
27	"thank you"
29	hi / hello
30	"good luck" "congrats"
31	age, social status, children
32	sports, sports teams
33	"me too" "same here"
34	my name is
35	weather
37	money/work
39	study/student/teacher/classes

Table 4.4: Answer clusters formed from the GTKY dataset using agglomerative clustering.

Training data format example

```
<|startofhistory|><|endofhistory|> <|startoftext|>hi , how are you doing to-
day ? i am spending time with my 4 sisters what are you up to<|endoftext|>
```

```
<|startofhistory|><|startoftext|>hi , how are you doing today ? i am spend-
ing time with my 4 sisters what are you up to<|endoftext|><|endofhistory|>
<|startoftext|>what do you do for a living ? i'm a researcher i'm researching the
fact that mermaids are real<|endoftext|>
```

The GTKY-preprocessed dataset was similarly preprocessed to have history tokens. However, as the dataset had been first automatically and then manually processed, there is no known start and end of a conversation, and as such the history tokens are included purely to teach the model the pattern.

GTKY training data format

```
<|startofhistory|><|endofhistory|> <|startoftext|>What is your favorite
foods? i like soups: kale-white bean-bacon is one of my favs.<|endoftext|>
```

```
<|startofhistory|><|endofhistory|> <|startoftext|>Have you been to the Ten-
nment Museum in NYC? i have not been to that museum.<|endoftext|>
```

The datasets were named PC-history and GTKY-history. Some variations were created of the history-focused datasets. One variation mixed in all of the questions and answers with history tokens from the GTKY-history dataset into the PC-history dataset. The data was mixed by iteratively taking one "conversation" from the GTKY-history dataset and then one conversation from the PC-history. This was done until the smaller dataset (GTKY-history) ran out of conversations, at which point the remaining conversations from the other dataset was inserted.

Mixing GTKY-history into PC-history

```
GTKY
PC
GTKY
PC
...
(GTKY-history out of conversations)
PC
PC
```

Similarly, the GTKY-history was merged with a few entries from the PC-history to double the size of the dataset. For every 4 GTKY-history "conversations" one conversation from PC-history was added. Note that the PC-history is 11.1 MB while GTKY-history is 166kB. The reason to merge was that the models were initially fine-tuned on the PC-history dataset, and then fine-tuned on the GTKY-history dataset, as it contains cleaner (manually processed) data. Since the GTKY-history does not contain a history of questions per conversation as it does in PC-history, it was hypothesized that the model may 'forget' to use history after being fine-tuned on the GTKY-history. Forgetting here refers to the model generating questions and answers, but it does not add previous questions and answers into the history section. Therefore, by mixing GTKY-history and PC-history, it is assumed the model does not 'forget' to use history.

A second variation extended the previous variation by including a limit on the number of past questions and answer pairs in the history section. The limit was arbitrarily set to 4 such that only the 4 most recent past pairs were kept in the history for the conversation.

The last variation used the Persona-Chat dataset without extracting questions and answers. Thereby the dataset contains the original conversation order with taking history into consideration. The PC-history dataset contains history of past questions and answers but some of the context may be lost between one pair and another due to the automatic extraction method. Therefore, the idea with the last variation was to initially teach the model a better context understanding and thereafter be fine-tuned for question answering.

Example of loss of context due to the extraction method

Past pair (history): no kidding ? of course . i love to listen to rock .

New pair: wow , does he live there or work ? live . moved there about ten years ago for a computer tech job .

4.6.2 Fine-Tuning new Generative Models

After creating new datasets, new models could be fine-tuned. An overview of the models is shown in table 4.5. First model #18 was fine-tuned 1000 iterations on the PC-history dataset and then 60 iterations on the GTKY-history dataset. The difference in iterations is due to the size difference between the two datasets. Model #19 was fine-tuned on 1000 iterations on the PC-history mixed with GTKY-history dataset and thereafter 250 iterations on the GTKY-history that was mixed with a few samples from the PC-history dataset. Model #20 was fine-tuned 1000 iterations on the Persona-Chat dataset with original context, mixed with GTKY-

Model	Fine-tuning description
#18	1000 iterations on the PC-history dataset, thereafter 60 iterations on the GTKY-history dataset.
#19	1000 iterations on the PC-history mixed with GTKY-history dataset, thereafter 250 iteration on GTKY-history mixed with a few samples from PC-history.
#20	1000 iterations on the Persona-Chat dataset (original context), mixed with GTKY-history.
#21	similar to #19 but with 4 question and answer pair limit in history

Table 4.5: Fine-tuning GPT-2 models with history

history. The last model was fine-tuned similarly to model #19 but with the 4 question and answer pair history limit. Each model was a 124M parameter model.

Additional fine-tuning experiments had previously been done after model #10 and before model #18, but these models were not documented.

4.6.3 Context Testing and Automatic Evaluation

Based on the fact that use of similarity ranking results in selection of more relevant answers (4.4.5), an automatic evaluator was created based on similarity scores. As such the metric has an inherent bias towards answers which are more similar to the question. The automatic method did not consider ties but in hindsight the concept could be implemented by using a Δ similarity threshold for how large the similarity difference should be to be considered a more relevant answer.

A test was held to compare whether using one message pair as history would lead to more relevant answers than if no previous history/context was given, when generating an answer to the next question. An overview of the resulting scores are provided in table 4.6. Two generative models ran in parallel to generate answers, one was given no message history in the input, while the other model received 1 question and answer pair history. Three results from three methods were saved:

- The “vanilla answer”
- Vanilla answer with ranking
- Answer with 1 message history and ranking

Method	Vanilla	Vanilla w/ ranking	History + ranking
Auto.	10	37	42
Auto. One-on-One	13	76	-
Manu. One-on-One	7	22	-
Auto. One-on-One	-	33	56
Manu. One-on-One	-	15	14

Table 4.6: An overview of the distributed points to three methods over internal automatic and manual tests.

Vanilla refers to the method as seen in the user test; no previous context given to the model and simply returning the top answer in the list of over-generated answers. For each answer from each method, there was a separate history embedding that was updated. Cosine similarity was then calculated between the answer and the answer’s history embedding, to serve as the metric for automatic evaluation. The answer with the highest similarity was determined as the winner for the current user question. The results when generating answers to 89 questions were [10, 37, 42], meaning that 10 points were given to the vanilla method,

37 points to the second method, and 42 points to the third method. When running one-on-one automatic evaluation on vanilla method against the second method, the score resulted in [13, 76], with 76 going to the second method. Manual evaluation was also performed, to confirm whether the automatic evaluation worked, which resulted in [7, 22], meaning that most questions tied. Automatic evaluation of the second method against the third method resulted in [33, 56] with 56 going to the third method. Manual evaluation resulted in [15, 14]. It was observed that using history sometimes led to repeated answers which affected the evaluation, meaning that the model generated, and through ranking selected the same answer that was seen in the answer to the previous question. One idea was therefore to only use history when observing that the question is a follow up question. Another idea was to remove repeated answers, see section (4.6.4).

Another test was performed where one model always got 1 message pair as history and another model got the entire history of the conversation as input. Automatic evaluation resulted in [36, 40] whereas manual evaluation resulted in [10, 15]. Although the model with the entire history scored better on manual and automatic evaluation (40 and 15), it was observed to generate more repeated answers.

A conclusion of the context testing is that providing the model with one message pair as history improves its chances of answering follow-up questions, although it may lead to repeated answers which affects the number of relevant answers. Therefore it may be necessary to have a way to identify follow up questions (4.5) such that history is only provided if the question is a follow-up question, to reduce occurrences of repeated answers, or attempt to remove the repeated answers (4.6.4).

4.6.4 Repeated Answer Removal

After introducing history that is given to the generative model, it was observed that the model would generate and choose an exact same answer that was already given in the past reply. This is more of an issue related to fine-tuning the model on more and better data than it is a ranking and answer selection problem. However, at this point there was no more time for creating datasets to train new models. Therefore, a function was created that takes in the past answer and all new generated answers and removes any answer that shares a majority of words with the past answer (majority determined as above 66.67%). The function does not take into consideration the word order, it simply tokenizes the answer and counts the number of shared tokens/words. If the shared number of words is above 66.67% then the answer is removed from the list of answers before the answers go through ranking.

4.6.5 Saving and Re-Using Past Messages

After training new models that take past questions and answers in a dedicated 'history' section, as part of input to the generative model, a new structure was made to save questions and answers. The question and answer classifiers were used to first classify the question and the answer, and then the information was stored in a pandas dataframe. The information stored was: the question, the answer, their respective classification labels along with their user id. The history that was saved included all questions and all answers, from any component (template, retrieval, generative). The generative component made use of the user history by first classifying a new question, and then retrieving up to the 4 most recent questions with the same question classification, along with their answers. Then, it also retrieved the question and answer that occurred just before the new question, regardless of its topic. If a question was classified as a follow-up question, then the question was labeled as the previous class when storing it in the user history.

Question classification example

Q1: What is your favorite sport? (sports/games)

A1: My favorite sport is football.

Q2: Why? (follow-up question type) -> (sports/games)

The history embedding was updated in turn, starting with the oldest question and answer in the history, until the new question. Then the input consisting of the history and the new question was given to the generative model to over-generate a number of answers, in this case 8. Then the answers were processed to remove occurrences of tokens. The answer processing function was updated to consider the new history tokens: `<|startfohistory|>` and `<|endfohistory|>`. The answers were then ranked, and the top ranked answer was returned.

The answer classification was never utilized in the agent due to time constraints, although it ideally could be used not only as part of the ranking or a pre-ranking filter but also to find contradictions with the help of Natural Language Inference (NLI) methods. The idea to find contradictions was to compare the generated answers with the past answers in the topic. If the agent said that it loves being outdoors and hiking, then in the future, generated answers that contradict this statement should be removed before ranking. Contradiction detection and avoidance is left as future work.

4.6.6 Model Selection

Four models were fine-tuned to consider history (#18, #19, #20, #21). Therefore, automatic and manual tests were performed internally in order to find which model to use in the system. As such, the points distributed in the tests were used solely for comparing the models to each other. The results of the tests are shown in table 4.7.

Eval. Type	Criteria	Dataset	#18	#19	#20	#21
Automatic	Highest ranked	User data	67	69	70	68
Manual	Acceptable	User data	85	76	64	98
Automatic	One-on-One	User data	131	-	-	123
Automatic	One-on-One	Clean user data	67	-	-	61
Automatic	Highest ranked	Generated	33	17	21	15
Automatic	One-on-One	Generated	52	-	-	34
Manual	Relevancy	Generated	22	-	-	17

Table 4.7: An overview of the distributed points to four trained models over different internal automatic and manual tests.

A script was made that ran the system on every conversation from the user test data by feeding the system the users' utterances. The script ran once for each model and the inputs (user utterances) and outputs (answers) were stored in a csv file. Automatic evaluation was then performed by, for each question, loading the answer given by the respective models. The four answers were then ranked through the existing ranking system to find which model's answer was ranked the highest. To rank the answers, a history embedding was necessary to find the relevance between the answer, the question and previous history. In this case, a history embedding was maintained by only updating the embedding on the questions instead of questions and answers. The automatic evaluation resulted in [67, 69, 70, 68], with the model order (18, 19, 20, 21). The automatic evaluation therefore found the models to be equally good.

A manual evaluation was done where, for each question, a point was given to the models whose answers were acceptable. This means that a point can be given to all answers if all of them are acceptable. An acceptable answer is considered one that does not contradict itself in the same message and is relevant enough to the question to where the answer seems natural. It does not take into consideration whether it contradicts the past message. On the data consisting of 239 questions, the scores were [85, 76, 64, 98]. In this case, if an answer was given by a component other than the generative model, then no score was given since the answer

is a template or from the retrieval database. If the “question” was not a question or if it was a too specific follow-up question, then again, no score was given. The manual evaluation showed that model 18 and 21 may be the best models, with 85 and 98 points respectively. Automatic evaluation was then performed again, by only considering model 18 and model 21, which resulted in 131 points to model 18 and 123 points to model 21. By instead skipping non-ideal questions (non-question user inputs and specific follow-up questions) and running automatic evaluation again, the score was 67 and 61 to model 18 and 21, respectively.

Since the user test data contained non-question utterances by users and specific follow-up questions, this in turn leads to poor automatic evaluation. Therefore, each generative model ran on a list of 86 questions, see appendix A.8, to generate answers. The questions had been previously extracted after a model was accidentally trained purely on question asking, in the beginning of the development. The automatic evaluation was performed as previously described, by ranking the answers from the different models, and giving a point to the model with the highest-ranking answer. The results were [33, 17, 21, 15] when considering all four, and [52, 34] when considering model 18 and model 21. Manual evaluation was done once again between model 18 and model 21, where the points are given based on answer relevancy. The result was 47 tied points, 22 points to model 18 and 17 points to model 21.

The first manual evaluation resulted in model 21 being the best whereas the following automatic evaluation showed that model 18 was better than model 21. The automatic and manual evaluation of the answers to the 86 questions resulted in model 18 being favored. Therefore, after all the tests and observing the generated answers by the different models, it was decided to use model 18. It should be noted that both automatic and manual evaluations can be further improved. The flaw of manual evaluation is that of human error and the possibility of inconsistent scoring. The flaw of automatic evaluation is that of missing out on semantic meaning and whether or not the answer makes sense [25].

4.6.7 Response Time Improvement

Response time from the generative model depends on the number of answers to generate, their length, and how many answers can be generated in parallel by the GPU (batch-size). Generating 8 answers took around 5 seconds on a laptop with a NVIDIA GeForce GTX 1650 graphics card. From the user test it was observed that the generative model took longer and longer to generate over time, seemingly correlating with the number of times the model was called, taking up to 32 seconds during one run. Therefore, the TensorFlow session associated with the model was henceforth restarted after being called to generate a set number of times (10) to avoid this phenomenon. The restart was observed to take around 3 seconds, sometimes a couple of seconds longer.

After testing the GPT-2 model implemented in Hugging Face’s library, it was found that the response time was faster, that it could generate answers from concurrent threads without loading duplicated models (3.9), and that the model did not take longer and longer after generating hundreds of messages. Therefore, these issues were deemed to be due to the gpt-2-simple library. Therefore, the GPT-2 source code from OpenAI was used as a base and adapted together with some extracted parts from the “gpt_2.py” script from the gpt-2-simple library. The resulting script was then able to generate without the previous issues. Generating 8 answers then seemed to consistently take less than 2 seconds on the GTX 1650, and closer to 1 second on a NVIDIA GeForce RTX 2060 graphics card. Concurrent generation, e.g. from two (or more) clients, was henceforth possible using one model. However, concurrent generation slows down the response time for both clients, as it can be compared to telling the model to generate 16 answers. As a sanity check, a test was made to see whether using duplicated models would improve concurrent response time, but no significant difference was observed. As such, there was no need for duplicated models with the new generative script.

4.7 Comparison Between Previous and Current Agent

The updated agent has a ranking system and conversation history, which should lead to higher quality responses from the generative component. When providing history to the model, it is able to have some context understanding. The following example shows an interaction where the agent is able to stay on topic even when the user asks non-specific questions, although it has problems with properly answering the questions. When asked "What kind?" the agent was unable to answer what kind of pies it bakes but it was able to remain on topic "Pie is a nice side dish.". When asked "which one is your favorite?" the agent answers with its favorite cake, which has some topic relevance (desserts). The updated architecture with answer ranking is illustrated in figure 4.4.

Context understanding in the new generative model

HUMAN: do you bake a lot of pies?

MACHINE: I bake a lot of pies, too.

HUMAN: what kind?

MACHINE: Pie is a nice side dish.

HUMAN: what kind of pies do you bake?

MACHINE: I bake a variety of pies.

HUMAN: which one is your favorite?

MACHINE: I don't have a favorite, but if I do have a favorite, I'd go with string cake

HUMAN: That's not a pie though!

MACHINE: It's a cake.

Data from the first user test was used to compare the updated component with the old generative component. The comparison was made on 28 follow-up questions, by generating new answers to the follow-up questions, while providing the original data as conversation history. It was observed that the new component more often outputs relevant responses compared to the original data. However, it still has problems with providing a relevant answer to the question. When rating which answer was more relevant to the question when comparing the original answer and the new answer, it was found that the new answers were more relevant 13 times, the original was more relevant 4 times and 11 answers were considered equally relevant or irrelevant.

Four samples from the user test data comparing original and new answers to follow-up questions

Q: Do you like Scotch?

A: I do like scotch. I also like tapioca pudding.

Q: What does tapicoa pudding taste like?

Original: It's got lots of pudding but a bit of chocolate chips.

New: Rich and creamy, I especially like the buttery, flavor.

Q: Do you like carrots?

A: I dislike carrots.

Q: Why do you dislike carrots?

Original: They're just too big and aren't golden brown.

New: My boyfriend loves carrots, but he doesn't really like me.

Q: Do you like chocolate?

A: I don't like chocolate but I like tapioca pudding.

Q: what is tapioca pudding?

Original: It's like pudding but without the chocolate chips.

New: i prefer cake with chocolate or cream.

Q: What is your favorite colour

A: I love hues of blue, pink, and orange.

Q: Why?

Original: Well, my dad and I spend a lot of time together, so it's definitely a bonding experience.

New: haha, I have no idea.

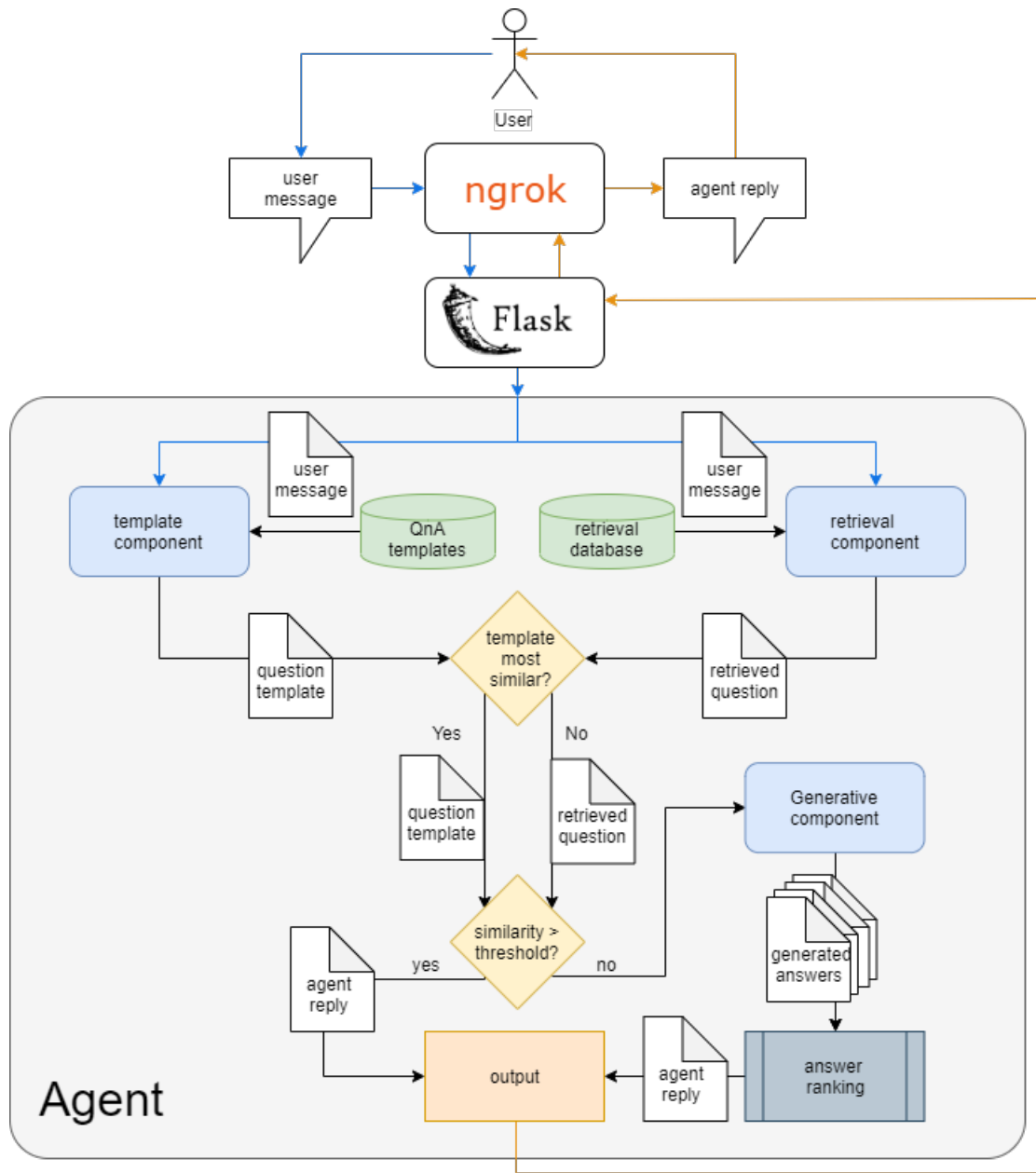


Figure 4.4: Updated architecture of the question answering agent

5 | Development of a Question Asking Chatbot

In order to make the conversation flow engaging, it is essential for the chatbot to have the ability to ask users questions. By asking questions to the users, it prevents one-sided conversations where the user is always the one who initiates the topic. It also shows that the chatbot listens and pays attention to the current topic, and is aware of the current discussion. By asking questions, it makes the whole conversation flow more realistic, because that is how a real world conversation between two friends would take place.

5.1 Types of Question

The first type of question would be opening questions, which are general and are asked frequently in everyday life when people meet. Such questions are for example, “How are you doing today?”. Additionally, there were two types of questions to be generated:

- Immediate follow-up questions.
- Questions that are related to the stored information about the user.

For the immediate follow-up questions such as *how*, *why*, *when* and so on, inspirations were taken from some question templates from [18]. If the current topic relates to, for example a sport *football*, several types of general follow-up questions can be generated. An example can be “What do you like about football?” or “Do you happen to play football?”, and the user answers can be stored and utilized when generating future questions.

The concept of memory was introduced in order to generate relevant questions after the first interaction. This was achieved by storing desired extracted information from the user into a dataframe. For example, if the topic *football* was mentioned in the first interaction, then one or several questions regarding this topic will be generated in the future, such as “Do you happen to also watch football?” or “Did you do it recently?”. The chatbot would then be able to “remember” information about the user, and would be able to perform conversations in a way similar to what people often do in real life.

The available topics were reduced down to two, as the main focus lies on investigating whether or not the use of memory can improve the user experience. It was sufficient to supply the chatbot knowledge within a few common topics. Sports and animals are the two major topics that was tackled on.

5.2 Rasa Framework

The project started off with building a simple prototype chatbot in order to understand how the framework functions, and to get a clear picture of the internal structure.

5.2.1 A Basic Chatbot

Rasa’s documentation provides a short tutorial¹ on building a first simple friendly chatbot. The story has two routes, it first asks the user how s/he is doing, and if the reply is positive, then it acknowledges; if the reply is negative, then it will send a picture of a tiger cub. The most important files are introduced here, which are:

- A *config.yml* file, which contains the configuration of Rasa NLU and Core models.
- A *nlu.md* file, which includes all NLU training data.
- A *stories.md* file, which includes all samples of conversations in simplified form.
- A *domain.yml* file, which defines the chatbot’s domain/knowledge.

A few additional important files are discussed in 5.6.

5.3 Rasa NLU

SpacyNLP² provides pre-trained language models, and various components from this library were used. Pre-trained word embeddings are helpful as they already encode some linguistic knowledge, and Spacy Featurizer provides pre-trained word embeddings from either GloVe or fastText in many different languages³. This is useful when there are few training data. For example, if a training example “I want to buy apples” is provided, and Rasa is supposed to predict a “get pears” intent, then the model would already know that the words “apples” and “pears” are similar. The model was constructed after defining a pipeline which specified the components it should use.

There were 3 pre-trained statistical models for English available, a small model *en_core_web_sm*, a medium model *en_core_web_md* and a large model *en_core_web_lg*. The medium model was used because its significantly smaller model size compared to the large one, 48 MB and 746 MB respectively. The small model had the least size with as little as 11 MB, but it was only trained on OntoNotes⁴ and does not contain word vectors. While the medium and the large models were both English multi-task convolutional neural network (CNN) trained on OntoNotes, and had GloVe vectors trained on Common Crawl⁵. Both had the ability to assign word vectors, POS tags, dependency parse and use named entities.

English was the language that the model would use, and was specified with “en”. The case sensitivity was set to *False* when retrieving word vectors, this decided if the casing of the word was relevant. For example, “Hello” and “hello” would retrieve the same vector. There might be other occasions when the casing matter and must be differentiated, in such case it should be set to *True*.

5.3.1 Tokenization

The Spacy Tokenizer⁶ was used to create tokens, and it was a part of the Spacy NLP. It segments a text into words, punctuations marks and so on, and creates *Doc* objects with the discovered segment boundaries. A *Doc* is a sequence of *Token* objects. Spacy has introduced a novel tokenization algorithm⁷, which gives a better balance between performance, ease of definition and ease of alignment into the original text input.

¹<https://rasa.com/docs/rasa/user-guide/rasa-tutorial>

²<https://spacy.io>

³<https://rasa.com/docs/rasa/nlu/choosing-a-pipeline/>

⁴<https://catalog.ldc.upenn.edu/LDC2013T19>

⁵<https://nlp.stanford.edu/projects/glove/>

⁶<https://spacy.io/api/tokenizer>

⁷<https://spacy.io/usage/linguistic-features#how-tokenizer-works>

5.3.2 Featurization

The Spacy Featurizer was used to create features for entity extraction, intent classification, and response classification, it was also a part of the Spacy NLP. It is a type of dense featurizer which allows saving memory, thus able to train on larger datasets. It creates features for entity extraction, intent classification, and response classification.

A Regex Featurizer was included by default to create a list of regular expressions defined in the training data format during training. A feature will be set and it contains information on whether an expression was found in the user's input or not for each regex. Then all features are fed into an intent classifier and/or entity extractor to simplify classification. Regex features for entity extraction are supported by the DIET (Dual Intent Entity Transformer) classifier⁸ component.

A Lexical Syntactic Featurizer was included by default to create lexical and syntactic features for a user's message to support entity extraction. It moves with a sliding window over every token in the message and creates features according to a default configuration⁹. It was also possible to configure what kind of lexical and syntactic features it should extract.

Finally, a Count Vectors Featurizer from Scikit-learn¹⁰ was used to convert a collection of text documents to a matrix of token counts. It creates bag-of-words representation of user message, intent, and response. This featurizer can be configured to use *word* or *character n-grams*. It is set to "word" by default, it counts whole words and use word token counts as features. This featurizer was used for a second time where it was set to "char_wb", which would look at sub-word sequences of characters.

5.3.3 Entity Recognition, Intent Classification and Response Selector

The Spacy Entity Extractor was used to predict the user input's entities. It uses a statistical BIOES (Begin, In, Last, Unit, Out) transition model¹¹. A potential disadvantage is that this component uses the built-in entity extraction models from Spacy and currently it cannot be retrained. Additionally, this extractor does not provide any confidence scores, therefore it is less flexible and customizable. On the other hand, there are several types of predefined entities that the models can recognize¹², including person, organization, location, product and many more. Here three types of entities were configured, which are "PERSON", "ORG" and "GPE". The reason that "ORG" and "GPE" are also included was that occasionally, certain personal names were identified as companies, agencies, institutions and such.

The DIET classifier could be used for both intent classification and entity extraction. The architecture¹³ is based on a transformer which is shared for both tasks. This classifier was used to extract custom entities that were present in the domain, such as sport and animal, while the Spacy's entity extractor was used to extract various named entities that were already supported by the models. There are several parameters that can be configured, "epochs" sets the number of times the algorithm will see the training data. As a starting point, it is set to 200 by recommendation. Depending on the situation, the model may need more epochs to properly learn, while other times adding more epochs do not influence the performance. The lower the number of epochs, the faster the model is trained. "ranking_length" is the number of top actions to normalize scores for loss type *softmax*, and it is set to 5 as recommended.

⁸<https://blog.rasa.com/introducing-dual-intent-and-entity-transformer-diet-state-of-the-art-performance-on-a-lightweight-architecture>

⁹<https://rasa.com/docs/rasa/nlu/components/#lexicalsyntacticfeaturizer>

¹⁰https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

¹¹<https://rasa.com/docs/rasa/nlu/components/#spacyentityextractor>

¹²<https://spacy.io/api/annotation#section-named-entities>

¹³https://www.youtube.com/playlist?list=PL75e0qA87d1G-za8eLI6t0_Pbxafk-cxb

An Entity Synonym Mapper component was used to map synonymous entity values to the same value. With this component, the model has the ability to detect and recognize similar entity values and return just one consistent value. This is useful as users may not refer to a certain object using the exact same name.

Finally, a response retrieval model was built by the Response Selector component. It was used to predict the chatbot's response from a set of candidate responses. It follows the exact same neural network architecture and optimization as the DIET classifier.

5.4 VADER Sentiment Analysis

The purpose of using sentiment analyser was to sort the extracted entities by their sentiment values. This leads to a ranking system and the chatbot will prioritize to generate questions related to the topics that have higher sentiment values. However, since the available topics have been narrowed down to *sport* and *animal*, this component is no longer required. Since the concept is expandable, it would be more useful if the chatbot handles a wider variety of topics, which can be implemented in the future.

5.5 Semantic Network

With ConceptNet, the number of API calls are limited, however this is not a major issue for the project since the chatbot will not be commercialized or be used publicly.

These relations could help the chatbot understand a given word, and various questions can be constructed based on these relations. For example, given an entity value *football*, the is-a relation provides a few suggestions, figure 5.1 shows top 3 results based on the weight. A relevant question such as "Do you like any other *game*?" can be constructed, although the word "sport" would have been the ideal choice.



Figure 5.1: Some relations provided by ConceptNet.

On the other hand, a disadvantage was that the information may not always be usable or as desired, because it is based on the information in the Open Mind Common Sense (OMCS)¹⁴ database, which is built on a large commonsense knowledge base from the contributions of many thousands of people across the Web. This results in potentially inconsistent or missing information and relations for a given search word. Therefore, this tool was not considered.

5.6 Rasa Core

Rasa Core handles the dialog management, it keeps track of a conversation and decides on how to proceed. It generates a probability model which decides a set of actions to perform based on the previous user inputs.

¹⁴<https://www.media.mit.edu/projects/open-mind-common-sense/overview>

5.6.1 Story

Several stories were supplied in order to cover a wide range of general opening and closing conversations, such as responding to the user's greeting, asking how the user is doing, and saying goodbye. Additionally, the chatbot was also taught to handle unexpected scenarios, for example when the user's input does not make sense, or is written in another language, or anything that the chatbot had not learned and therefore was not able to give a proper reply. The solution was to respond that the chatbot did not understand the user's input, or to ask the user to rephrase the sentence. These stories are usually simple and short, and do not contain more than two responses or actions. Below are a few examples of such stories.

A few examples of stories for common opening and closing conversations

```
## story - thank
* user.thank
- utter_reply.to_thank
```

```
## story - sad
* user.sad
- utter_reply.to_sad
```

```
## story - good
* user.good
- utter_reply.to_good
```

```
## story - bye
* user.bye
- utter_greet.bye
```

```
## story - reject
* user.reject
- utter_reply.to.reject
- utter_greet.bye
```

After the chatbot was capable of handling the common opening and closing conversations, a few more complex stories were added to take care of conversations that involve more details and topics related to sports and animals. When the chatbot is deployed on a virtual machine later on and open for user interaction, more stories can be generated from the conversation history and used to retrain the chatbot.

5.6.2 Domain

The domain was essentially the universe that the chatbot lives in. It started with defining all the intents that the chatbot was able to classify, such as *user.thank*, *user.sad*, *user.good*, *user.bye* and *user.reject*, which were seen previously in 5.6.1. Multiple intents could be created for various topics, such as *discuss_sport* for sport related discussions, and *discuss_animal* for animal related discussions. However, the more intents created, the more training stories would be required. Below are two example stories which train the chatbot to ask follow-up questions.

Example stories for topic related to sports and animals

```
## story - sport topic
* discuss_sport
- utter_ask_why_sport
```

```
## story - animal topic
```

```
* discuss_animal
- utter_ask_why_animal
```

Instead of creating a unique intent for each topic, a better approach was to generalize and group them as one intent, in this case called *enter_data*. It consisted of all NLU training examples for topics related to sports and animals. With this approach, fewer intents would be needed, and the design of training stories was simplified and more generalized. Below are a few examples of NLU training data about sport and animal topics, as well as a more generalized story that trains the chatbot to ask the same follow-up question.

A few examples of NLU training data for *enter_data* intent

```
## intent:enter_data
- my favorite sport is [football](sport)
- Well, I do a lot of [yoga](sport)
- i like playing [tennis](sport)
- i would say [skateboarding](sport)
- [cats](animal) are my favorite!
- i'm a [dog](animal) person
```

One story to cover topics related to sports and animals

```
## story - discuss topic and follow up with asking why
* enter_data
- utter_ask_why
```

In the NLU training data, the entity types were defined using brackets. Since the focused topics were sports and animals, the two main entities are then (*sport*) and (*animal*). The entity's value was defined using square brackets, such as *[football]* and *[dog]*. This helps the chatbot to map these values to the desired entities. For example, if the user inputs "I like monkey.", then the chatbot is able to extract entity animal with value *monkey*. Similarly, if the user inputs "I really like basketball.", then the chatbot is able to extract entity sport with value *basketball*. The extracted entities were stored into slots. Slots are essentially the chatbot's memory, and is covered more in detail in 5.6.3.

5.6.3 Slot

The chatbot was equipped with slots, which are its short-term memories. It was considered as short-term memories because the data would be lost after each session. When the chatbot restarts, its slots are emptied. To implement long-term memories, certain slots are saved to an external database. Each memory slots stores a piece of information. With the help of slots, the chatbot was able to memorize certain information throughout the conversation, such as the user's name.

The numbers of slots depended on how many pieces of information the chatbot needs to remember. Slots such as *email* and *name* were commonly used and were essential to have in order to have a general knowledge base of the user. Other types of information to be stored should be usable in a follow-up and/or long-term question. For example, if the user inputs a favorite sport to be *basketball*, then the value "basketball" is stored in a unfeaturized slot named "type_of_sport". The chatbot can then utilize this information and ask a follow-up question such as "Did you play *basketball* recently?". The answer is expected to be a *yes* or *no*, this is converted to a True or False value, and stored in a boolean slot named "recent_active_sport". Now the chatbot knows that whether or not the user has done the activity recently. During the next session, the chatbot can then utilize this information to ask related question, depending on the boolean value. For example, if the boolean returns True, the chatbot can ask how the activity went; if the boolean returns False, the chatbot can ask if the user will perform the activity in the near future. Below are a few examples of slots

that store such information that can be utilized to ask follow-up questions and/or long-term questions.

A few examples of slots that store useful information that can be utilized to ask follow-up questions and/or long-term questions

```
type_of_animal:
type: unfeaturized
own_animal:
type: bool
animal_breed:
type: unfeaturized
```

More slots would be needed as the numbers of topics increase, and the deeper each conversation the chatbot tries to cover. Therefore, limiting the topics down to as few as two was beneficial, since less work would be needed for the conversation flow design. The numbers of slots depend on how much information is desired and to be extracted. Certain information from the slots will be stored externally in a dataframe, based on the design of long-term questions. For example, if the user owns an animal, and the animal is a dog, then a potential long-term question can be “So what breed is your dog?” or “What color is your dog?”. In order to ask such question, certain information such as that the user owns an animal and the type of animal is a dog, are essential to store. If more follow-up questions are required, additional slots such as “animal_color” and “animal_breed” can be added. Essentially, the more information is extracted, the more slots will be required. Below are the slots used for topics related to sports and animals, they were 14 slots in total.

Slots used for topics related to sports and animals

```
type_of_sport
reason_of_like_sport
recent_active_sport
how_go_sport
play_in_future
watch_sport
type_of_animal
own_animal
animal_breed
plan_to_own_animal
animal_color
animal_size
how_old_is_animal
reason_of_like_animal
```

There were 6 slots for the sport topic, where 2 slots “type_of_sport” and “recent_active_sport” were stored externally; and 8 slots for the animal topic, where 2 slots “type_of_animal” and “own_animal” were stored externally. These were stored externally in order for the chatbot to memorize some personal information about the user, therefore it was able to ask personalized question in the next conversation. For example, if the user’s favorite animal is cat, and s/he happens to own one, then based on these information, the chatbot is able to ask the cat’s breed, color and size; on the other hand, if the user’s favorite animal is cat but does not own one, then the chatbot can ask if s/he plans to own one in the future. Many more such follow-up and long-term question could be designed to make the chatbot more friendly and engaging.

5.6.4 Response

Responses are the messages/utterances that the chatbot is able to use to reply to a user. These responses were defined in the domain file. Below are a few examples for utterances *utter_reply.to_thank* and *utter_greet.bye*, which are seen previously in 5.6.1.

A few responses available for different utterances

utter_reply.to_thank:

- text: "No problem!"
- text: "Not at all!"
- text: "Any time!"

utter_greet.bye:

- text: "Ok see ya!"
- text: "Buh-bye."
- text: "Until next time!"

Each utterance can contain one or more candidate responses (templates), the chatbot will select one randomly. With this approach, the specific texts do not need to be provided when designing the training stories. It was sufficient to supply the type of utterance that was suitable given an intent. When the chatbot has classified a particular intent, it would know which utterance to use and picks one of the responses to send.

The chatbot's memory can be utilized and make the responses more dynamic. For example, when the user inputs "I really like basketball.", a follow-up question can be "What do you like about basketball?". However, the amount of responses will be enormous as there are many types of sports, and here is where slots are useful. The chatbot is able to extract the entity sport with value *basketball* from the above user input, it then stores this value into a slot named *type_of_sport*. The response can utilize the information from this slot, and a general follow-up question is adjusted to "What do you like about {type_of_sport}?". This response changes according to the extracted entity value from the user input. Now the user is able to talk about all kinds of sports, and the chatbot is able to follow up with a related question using just one or a few templates.

5.6.5 Action

Actions are similar to responses, except they have more functionalities. The chatbot can use responses to reply to the user, but it is a simple text replies, it can be customized with the help of slots to make the replies more dynamic, and that is about all to it. On the other hand, the chatbot can use actions to perform more complicated tasks, such as calling external API and storing and retrieving information from external dataframe.

To implement the chatbot's memory, an external dataframe was used. Table 5.1 shows the information that are stored externally in an excel file.

user	name	sport	recent_active	animal	own
REDACTED_EMAIL	REDACTED_NAME	football	false	cat	true

Table 5.1: Information that are stored externally in an excel file.

There are several slots that were filled with information, six of them were useful for generating long-term questions. Since there would be two conversation sessions, where the first session the chatbot would retrieve information, and use some of the information to generate personalized questions in the second session, it needed a way to identify and distinguish between new and existing users. The *email* slot was stored to the *user* column for a simple identification purpose and checking if user exists in a dataframe to distinguish between the first and second interaction. The *name* slot was stored to the *name* column, so that the chatbot

had the ability to refer the users by their names. The other four were related to sports and animals topics. At the end of first interaction, all slots were cleared, and information from these six slots were written to an excel file. During the start of the second interaction, when the user enters his/her email address, it will be used to check whether or not the same email is found in this excel file. If the email address is found, that means the user has interacted with the chatbot previously; otherwise, it means that it is the user's first interaction. A different story path will be used depending on if it is the user's first or second interaction. Below is an example when the user enters a second session.

Example of a beginning of a user's second session

User: hi
 Chatbot: SYSTEM: PLEASE enter your email address to continue.
 User: REDACTED_EMAIL
 Chatbot: Good day! How are you?
 User: Doing well, you?
 Chatbot: I'm doing pretty well.
 ...

Custom action such as calling API was also possible. For example, it is possible to get data from ConceptNet in JSON format, and various information can be retrieved depending on the relations that are looked for. In general, this helps to widen the chatbot knowledge and information regarding a specific topic, depending on how knowledgeable the chatbot needs to be. An extra functionality is tested, which enables the chatbot to send a URL where the given search word can be found, to emphasize that the chatbot remembers the previously mentioned topic. It can reply for example, "By the way, I saw this post the other day, check it out!" and insert the URL that is related to the topic. This is similar to a recommendation system. However, this was not implemented in the final version, as not every search word will have available data, although it might make the chatbot more interesting to interact with. Other API calls can be useful depending on what functionalities the chatbot was designed to have.

5.7 Rasa X

After a first basic simple model that was sufficient to cover the most common and general conversation scenarios, it was moved onto conversation-driven development using Rasa X. Conversation-driven development allows the chatbot learn from the users, and use those new messages as training data to retrain and improve the chatbot. Rasa X was essentially a layer on top of Rasa Open Source, and it was deployed on a virtual machine on Google Cloud¹⁵.

The virtual machine was a standard type with 2 CPU, 7.5 GB memory and no GPU. This met the requirements¹⁶ to host and run Rasa X. The operating system was Ubuntu 18.04¹⁷. Once the virtual machine was set up, Rasa X was installed using Docker Compose¹⁸. This was a good choice if the user traffic was not expected to be a large amount¹⁹, i.e. less than hundreds of concurrent users, which was the case for this study.

Rasa X has an Integrated Version Control feature, which automates data synchronization with a Git repository, annotates new data and pushes those changes with Git. The project files were uploaded onto GitHub. To connect Rasa X to the remote repository, a public SSH key was generated and added as a deploy key with write access on GitHub. After the connection

¹⁵<https://cloud.google.com/compute>

¹⁶<https://rasa.com/docs/rasa-x/installation-and-setup/install/docker-compose/#docker-compose-requirements>

¹⁷<https://releases.ubuntu.com/18.04>

¹⁸<https://docs.docker.com/compose>

¹⁹<https://rasa.com/docs/rasa-x/installation-and-setup/installation-guide>

had been established, Rasa X could pull from the repository and train the model. For any further adjustments, Rasa X could push the changes to the repository on the master branch directly, or to a new branch and perform a merge later.

5.8 Architecture

Figure 5.2 shows the architecture of the chatbot built using the Rasa framework. The chatbot was built using the open source Rasa framework, which consisted of two main components: Rasa Core and Rasa NLU. Rasa Core was the dialogue engine and the domain file defined the settings for the chatbot, such as what it should understand, what it could use to reply and so on. Rasa NLU took in NLU training data and conversations examples in order to train the chatbot, it could then take care of intent classification and entity extraction. The conversation flow design, and other functionalities such as storing and fetching information externally from a dataframe, were defined in the action file. This chatbot was deployed on a virtual machine with the help of Rasa X tool, which helped improving the chatbot as it interacted with users.

The conversation flow started with asking for a mean of identification, in this case the user's email address. The chatbot used the given email and check against a dataframe to see if the user exists to decide if it was the first or second interaction. The chatbot started the conversation with five chit chat questions, such as "How should I call you?". Thereafter, the chatbot proceeded to the main topics, which were related to sports and animals. In the first session, there were three questions related to sports, one opening question and two follow-up questions; and there was a total of four questions related to animals, one opening question and three follow-up questions. In the second session, there were three long-term questions related to sports; and four long-term questions related to animals. Not every single question would be asked, it depended on certain replies from the user, ones that return boolean values. Finally, the chatbot would thank the user for the participation and attach a link to the survey.

5.9 User test - Question Asking Chatbot

The purpose of the test for the question asking chatbot was to see how the user would interact with the chatbot, and to find out how the chatbot was perceived when it showed the ability to remember the users. The aim of the user test was to also reveal any potential faults in the system. More importantly, it should improve the design of the survey and the questionnaire in order to receive enough data and feedback to answer the research questions. The users were asked to interact with the chatbot for a first session and fill in a survey, then reset the chatbot and interact with it again for a second session, and finally fill in the same survey which contains the exact same questions. The session reset simulated the passing of time.

5.9.1 Survey

The interaction time for both sessions was estimated to be about five to seven minutes. After each interaction, the user was asked to fill in a survey that consists of five questions, two of which are the same from the question answering chatbot survey, in order to have some common metric for comparison. The users were asked to rate on a 5-point Likert scale from 1-"Completely disagree" to 5-"Completely agree". The five questions are shown below:

- The grammar of response is correct? (Grammar)
- The response is appropriate to be made in a normal conversation? (Appropriateness)
- The response is related to my text input? (Relevancy)
- The chatbot was engaging/interesting to interact with? (Engagingness)

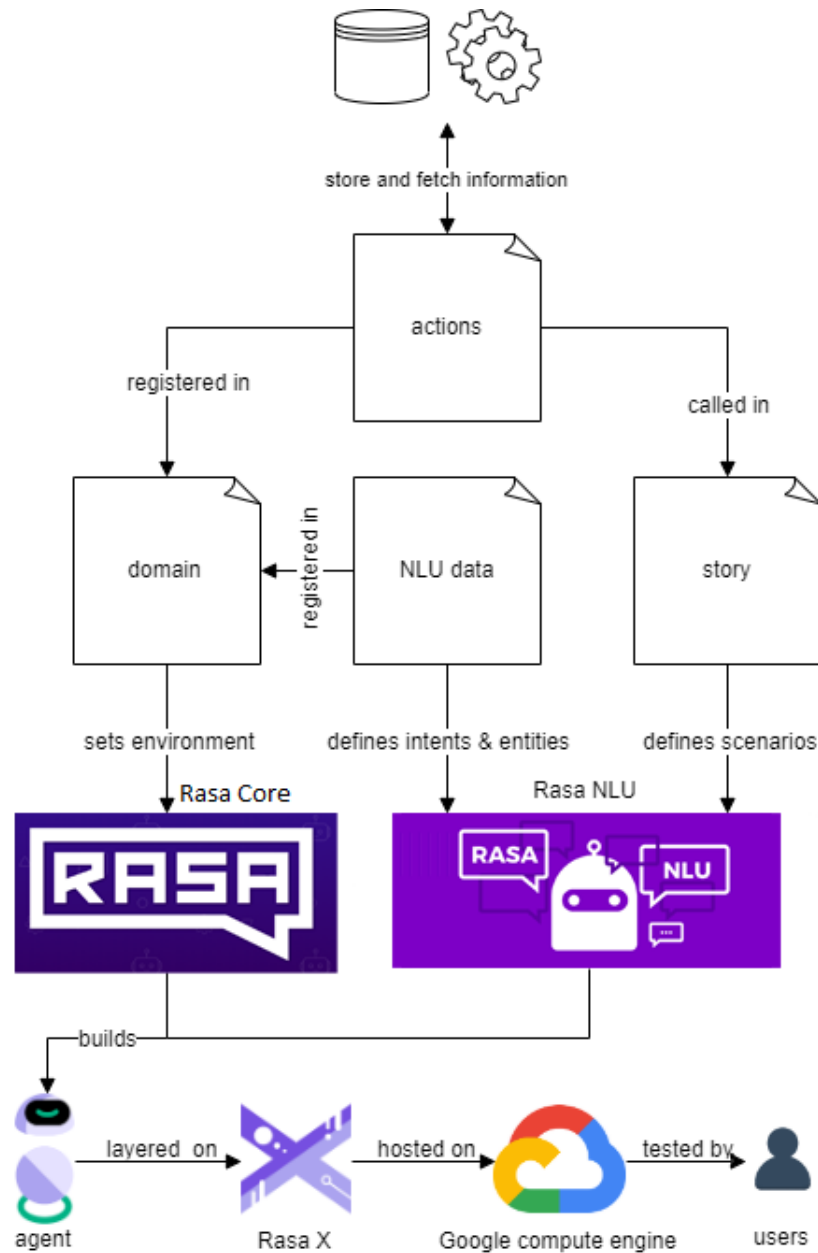


Figure 5.2: Architecture of the chatbot built using the Rasa framework, both the Rasa Core and Rasa NLU are used for dialog management and natural language understanding. The chatbot is hosted on a virtual machine on Google compute engine for user tests.

- It feels as if I'm chatting with a real person? (Realness)

5.9.2 Results

The user test had five participants in total, one of which was working as a PhD student, one was a recent PhD graduate, one was a recent master graduate, and the other two were in their master's degree study. Further investigation was conducted if one or more questions have low scores. Table 5.2 shows the mean results for each question.

As seen in the table, the results of the chatbot's grammar, appropriateness and engagingness from both sessions were consistent. However, there was a decrease in its relevancy and

	Session 1	Session 2	P-value
Grammar	4.4 (0.489)	4.8 (0.400)	0.242
Appropriateness	3.8 (0.400)	3.6 (1.019)	0.724
Relevancy	4.4 (0.489)	3.8 (1.166)	0.371
Engagingness	3.2 (0.400)	3.2 (0.748)	1.000
Realness	3.6 (0.489)	2.6 (1.019)	0.115

Table 5.2: Survey result of the first user test with the question asking chatbot, illustrating the mean value for each question and the p-values of t-tests.

realness. Therefore, further investigations and discussion were made with the users who had entered a lower score for these two metrics in the second session.

The first immediate shortcoming was on the survey itself. There lacked a clear instruction to inform the users on what the chatbot does and its limitation, as it mainly generated question, and the focus was on the usage of memory. This was not explicitly stated, therefore users might have different expectations and might take unexpected actions such as trying to ask the chatbot questions.

The second shortcoming was the opening chit chat questions. The intention of these small talks was to make the chatbot friendly and let the users feel more relaxed before jumping into the main topics. While some of the common chit chat questions worked as intended, other might raise the user's expectation on the chatbot's capability. Therefore, users might incline to test the chatbot's limitation and see what it could do, or reply with an answer that was not logical or suitable compare to a normal conversation between two people. Below is a sample of interactions from a user during both sessions, where questions were asked by the user, which was unanticipated.

A sample of the interactions from a user's both sessions

First session

Chatbot: Now, what's one sport that you like?

User: football

Chatbot: Interesting! What do you like about football?

User: I only like watching football

Chatbot: Did you do it recently (<30 days)?

User: no

Chatbot: I see. Another question, what's your one favorite pet?

User: fish

Chatbot: That's cool! Do you happen to own one?

User: no

Chatbot: Alright. Do you see yourself getting one in the future?

User: I am planning to buy one but not sure what to buy, maybe a shark, what do you think

Second session

Chatbot: I remember that you were interested in football right. You think you will try it out in the future?

User: what's your suggestion?

Chatbot: Do you happen to also watch football?

User: I have told you last time

Chatbot: We were also talking about fish last time. What do you like about it?

User: They are dumb and they taste good

Chatbot: I see. If you get to keep one, would you prefer a big one or a small one?

User: I prefer a small one

Chatbot: Great, it was nice talking to you!

Below is another sample of interactions from a user during both sessions, which is a better representation of a normal conversations between two people. In this case, the user still rated a lower score on the realness metrics. The main reason was that the user referred to the dog as a “she”, but the chatbot still used “it”, which showed that the chatbot was not able to comprehend fully. This could be improved by introducing an additional slot, however since it did not contribute significantly to answering the research question, this part was left out.

A sample of the interactions from another user’s both sessions**First session**

Chatbot: How should I call you?

User: REDACTED_NAME

Chatbot: Alright! How are you today?

User: I am fine, Thanks

Chatbot: I see, and where are you from originally?

User: I am from REDACTED_NAME

Chatbot: Mh I’m from the clouds. Do you happen to like American food?

User: Not so much

Chatbot: Haha ok, and what do you do for a living?

User: I am working in an university, as a teacher

Chatbot: Now, what’s one sport that you like?

User: Runing

Chatbot: Cool! Why do you like Runing?

User: I can do it with nobody, I mean, I dont like play sports with others

Chatbot: Did you do it recently (<30 days)?

User: Yes, sure

Chatbot: I see. Another question, what’s your one favorite pet?

User: dog

Chatbot: That’s cool! Do you happen to own one?

User: Yes, I do!

Chatbot: I see. So what breed of dog do you have?

User: It just a normal Chinese dog, I dot really know her bread, sorry

Second session

Chatbot: Welcome!

Chatbot: Haven’t seen you since last time we talked. How have you been?

User: Thanks for asking, I am pretty good

Chatbot: Alright! What did you do in your free time?

User: Play some video games

Chatbot: I see, and where do you currently live?

User: I live in REDACTED_NAME

Chatbot: How is the city?

User: Its a good city, I like it. Its clean and quiet

Chatbot: And how is the weather there?

User: The weather is good, not so warm and not so cold.

Chatbot: So last time we talked about Runing, and you did that recently. How did it go?

User: Its a bit hard to do it now, because its summer and I feel too warm to run

Chatbot: Do you happen to also watch Runing?

User: No

Chatbot: I remember that you have a dog. How old is it?

User: She is 3

Chatbot: And what color is it?

User: Black she is

It was unavoidable for users to also ask question, which the bot was not designed to handle the more complicated ones. However, this problem would be eased in the next stage when this chatbot is combined with the question answering chatbot.

Some of the chit chat questions had also risen the users' expectation on the chatbot's capability. Chitchat questions such as "How are you today?" were considered normal opening questions, on the other hand, questions such as "what do you do for a living?", while still belong to chitchat, the scope had narrowed down to topics related to jobs. This could mislead users that the chatbot was able to handle conversations related to jobs. Therefore, several chitchat questions were removed.

Out of context messages were treated normally. For example, when the first user answered to have a shark as a pet, the chatbot treated the animal as a normal house pet, which was unrealistic. This happens when users try to enter unrealistic or out-of-context answers, which do not happen in a normal conversation. The chatbot was not designed to handle such situations. However, this would not be a severe problem when the users answer the questions normally.

6 | Integration

The final step of development consisted of combining the question answering agent with the question asking agent into one system. The question asking agent was designed to have more control of the conversation flow and therefore it was decided to use this agent as the base for the system. The question answering agent was turned into a callable API where new rules had to be constructed as to when to make an API call from the base agent.

6.1 API Development

An API was created to be callable from the question asking component to be able to integrate the two components into one system with both question asking and question answering capabilities. The API source code and files were made publicly available under an MIT license¹. The question answering component was extended with an additional two components as a result of merging the systems. The components were developed with the basis of relying on the sentiment memory for the sake of self-disclosure. The first new component was developed to be called after a user has answered the opening question of a topic. The component then takes the answer, extracts the subject of the topic and compares the subject with the sentiment memory and returns an answer based on sentiment.

Conversation flow example

Q-chatbot: "What is your favorite animal?"

User: "I love cats!"

A-chatbot: "I like cats too, but I prefer dogs."

The second new component was developed to introduce a new topic by letting the agent mention its favorite subject in the topic and asking the user about theirs, see figure 6.1.

A-chatbot: "My favorite sport is football. Do you have a favorite sport?"

The API consist of three callable functions:

- The question answering function, the function takes user ID and user utterance as input and returns an answer.
- Self-disclosure function, the function takes user ID, user utterance and topic as input. This function is called after a user answers an initial question in some topic (once per topic). The agent then acknowledges and shares its own preference on the topic.
- Disclosure and reflect function, the function takes user ID, user utterance and topic as input. The user utterance is only given to be saved in the agent's conversation history. The function returns an output solely based on the topic. The function is called only once for each topic that has not yet been discussed.

¹<https://github.com/lcebear/memoryDialogueBot>

```

My favorite animal is sheep. Do you have a favorite animal?
> animal
My favorite animal is sheep. What do you think about sheep?
> sport
Do you like soccer? It is my favorite sport.
> sport
My favorite sport is soccer. How about you?
> sport
Do you have a favorite sport? Mine is soccer.
> sport
What is your favorite sport? I think mine is soccer.
> sport
My favorite sport is soccer. Do you also like soccer?
> serie
Do you like Better Call Saul? It is my favorite tv show.
> genre
Do you have a favorite genre? Mine is historical.
> music
What is your favorite music? I think mine is dance music.
> hobby
I like playing darts the most out of all hobbies. What do you think about hobbies, do you have a favorite?
> art
What is your favorite music? My favorite has to be dance music.
> interst
My favorite interest is to play games. Do you also like to play games?
> book
My favorite book is The Lord of the Rings. Do you have a favorite book?
> book
What is your favorite book? My favorite has to be The Lord of the Rings.
>

```

Figure 6.1: Illustrating the disclosure-and-reflect component, given topic as input.

The inputs and outputs to the API are in JSON format. The output from the API consists of an answer field, an error field, and a component field. The answer field is the answer returned by the question answering system. The error field is to inform whether an error occurred on the side of the question answering system. The error field is 'None' by default and therefore, an answer should only be considered if the error field is 'None'. The component field is returned purely to record which component of the system was used to get the response.

6.2 Integrated System Rules and Conversation Flow

The first necessary change when integrating the two components was to ease up the control of the conversation that the question asking agent has by allowing the user to ask their own questions before the agent asks another question.

Example flow of question asking agent

Q-chatbot: What is your favorite animal? (Opening question for topic)

User: I like cats. (Answer)

Q-chatbot: Do you have one as a pet? (follow-up question in topic)

Suggested flow for the integrated system

Q-chatbot: What is your favorite animal? (Opening question for topic)

User: I like cats. (Answer)

A-chatbot: I like cats too, but i prefer dogs. (Acknowledgement and self-disclosure)

In the suggested flow, the user will have two options for their next reply, either ask a question, "Why do you prefer dogs?", or acknowledge the agent's answer, "Ok.". If the user asks a question, the API will be called to answer it, otherwise the agent will continue with asking questions. Once the user starts asking questions, the user can be said to have control of the conversation and therefore rules have to be determined as to when the system should retake control and start asking questions again.

One possibility is to keep calling the question answering agent for as long as the user is asking questions. Once the user replies with a message that is classified as “Acknowledgement”, the question asking agent will retake control.

Another possibility is to allow the user to ask one question by calling the API, getting the answer and adding a question at the end of the answer such that the next agent response consists of an answer and a new question. This option was chosen in this case due to the scripted nature of the question asking agent. Only allowing one question by the user ensures that the conversation remain on topic. With the other option, the user may ask questions which lead the conversation into another topic, and once the agent gets control again, it asks a question about the original topic again, which may seem out of place.

Another addition is, for every user message, a check is performed to see whether or not the message intent is classified as a question. If it is, then an API call will be made to the question answering chatbot. This will handle cases where users add a question at the end. Below is an example where a user asks a question during the opening question phrase.

User is now able to ask question anytime during the conversation

Chatbot: Good day REDACTED_NAME! How are you?

User: Good day to you too. I am fine. A bit stressed. What about you?

Chatbot: Mostly I’m doing OK, just fine.

6.3 Final User Test

The final user test was made available online for a two-week period. A difference from the question answering chatbot’s user test (3.10) was that the test followed a more scripted/defined structure, which reduced the variability between user interactions. Subjects were told to interact with the agent in two sessions and fill in a survey after each session. The sessions were held on different days. The reason for having two sessions is to measure how the user perceived the question asking agent and its memory of the user.

For the survey, participants were told to first fill in their age, email, English proficiency and to answer a few questions. The questions were:

- Have you interacted with a chatbot previously?
- Have you interacted with a chatbot in this study before?

Then, if the participant had interacted with a chatbot in this study before, they could fill in which chatbot they had interacted with (mark all that apply).

Then the participants were provided a link to where they could interact with the chatbot, as well as brief instructions. The chatbot was trained on a set of conversation flows (rasa related) and required that the user first sent a greeting message.

- Start the conversation by sending a "Hi" message.
- During the conversation, the chatbot will ask you a few questions; you will also have opportunities to ask the chatbot some questions as well.

After interacting with the chatbot, the participants were told to rate a number of statements on a scale of 1-5 from "Not at all" to "Very much". Three statements measuring the overall agent, two statements each for the question answering agent and question asking agent.

- The chatbot is engaging/interesting to interact with. (engagingness)
- It feels as if i’m chatting/talking with a real person. (realness)
- I would like to interact with the chatbot again in the future. (retention)

- The chatbot's answers are related/relevant to my questions. (answer relevancy)
- It feels as if the chatbot has its own personal background and personality. (persona)
- It feels as if the chatbot remembers (things about) me. (user memory)
- The chatbot's questions are related/relevant to my inputs. (question relevancy)

Additionally, participants were told to rate on a scale from 1 to 10:

- The chatbot's grammar from poor to excellent.
- The conversation flow from unnatural to natural.
- The quality of the chatbot's questions from unsatisfactory to satisfactory.
- The quality of the chatbot's answers (non-question replies) from unsatisfactory to satisfactory.

The survey for the second session was the same except that there were optional text fields to provide feedback if the participant gave a low rating on one measure and wanted to provide context as to why, or if the participant had any other feedback. The questions regarding age, English proficiency and previous experience with chatbots were not asked in the second survey, as they were already provided in the first survey, and were not expected to change.

Results

Issues with the test, as observed during it, was that users experienced difficulties with starting a session, mostly the second session. This was due to a number of reasons, either the user did not start with a greet message to initiate the agent's trained conversation flow or the agent was unable to identify the user, or the agent's session was not reset between first and second sessions. The agent was unable to identify the user if either the email was not provided to the agent, or if the user accidentally used different e-mails, or when there is a typo/accidental capitalization present. The agent's session not being fully reset automatically after 60 minutes was a technical error due to an incorrect setting in Rasa. Remaining information that are still left in slots are carried over to the next session, therefore the chatbot may skip asking certain questions if the corresponding slots are filled. It is uncertain as to whether the setting was incorrect from the start of the user test, or if the setting was lost after restarting the system on Monday the 20th of July. The session not restarting led to users being unable to start the second session unless they hit a reset button, which they were not provided instructions for.

The number of participants that interacted with the agent and filled in the first survey related to the first session was 17, out of which 5 had interacted with a chatbot from this study previously, and 5 of which had not interacted with any chatbot previously. The participants had a mean English proficiency value of 4.42. The age of the participants ranged from 18 to early 40 with a mean of 27.5. Out of the 17 participants, 3 were discarded. One of the discarded participants did not interact with the agent but still filled in the survey, another participant bypassed the agent's trained conversation flow, and the last participant was an outlier. The participant was an outlier due to interacting with the agent after the session was officially over, with a total of 25 messages during the entire interaction. The average number of messages were 8.375 (between 6-10 messages), where two of the messages include greeting and providing e-mail for identification. Out of the original 17 participants, 10 participants interacted with the agent a second time, 2 of which were unable to interact with the second session of the agent and were therefore discarded. One more participant was discarded due to filling in the survey over one week after interacting with the agent, which is not considered reliable results. A reliable result is considered to be one that is taken as soon as possible such that the subject still has the interaction in memory, e.g. directly after the interaction. An additional two participants were discarded due to faults in the chatbot where it had failed to extract the user's favorite sport in the first session, resulting in a template response asking

about "fail". The template response should not have been retrieved in this case. Therefore, 5 participants were considered for the second survey.

Consequence of failing to extract entity

Chatbot: Last time you told me about your favorite sport, what was it again? I forgot.

User: Slacklining

Chatbot: Last time we talked about fail, did you do it recently (<30 days)?

User: No

Chatbot: You think you will do it in the near future?

User: Hopefully not

The results are shown in table 6.1 and 6.2, where the first group "Session 1 (All)" represent the result from the 14 remaining participants, and the remaining columns represent the first and second session of the 5 participants that interacted with the agent's two sessions. The main results in the tables show a decrease in engagingness and conversation flow from session 1 and session 2. As well as an increase in realness, persona, question quality and user memory (due to long-term memory question asking). T-tests indicate that the only significant difference between the two sessions was the measure of user memory ($p=0.008$). The results are put in context in chapter 7 with tables comparing all three chatbots.

Group	Session 1 (All)	Session 1	Session 2	P-value
Engagingness	3.571, (0.852)	3.600, (0.548)	3.000, (0.707)	0.172
Realness	2.786, (1.122)	2.400, (0.548)	2.800, (0.837)	0.397
Retention	3.429, (0.756)	3.600, (0.548)	3.400, (0.894)	0.681
Persona	3.071, (1.072)	2.600, (0.894)	3.200, (1.483)	0.461
A-Relevancy	3.714, (1.490)	4.200, (0.837)	4.200, (0.837)	1.000
User Memory	3.286, (0.914)	3.000, (0.707)	4.400, (0.548)	0.008
Q-Relevancy	3.857, (1.406)	4.400, (0.894)	4.200, (0.837)	0.724

Table 6.1: Mean and standard deviation of the survey results from the final user test which used a 5-point Likert scale. Session 1 (All) represent a group of 14 participants and the remaining columns represent a group of 5 that completed both sessions.

Group	Session 1 (All)	Session 1	Session 2	P-value
Grammar	8.071, (2.093)	7.600, (2.302)	7.800, (1.483)	0.874
Flow	7.143, (1.512)	7.200, (0.837)	6.400, (2.881)	0.567
Question Quality	6.643, (2.274)	6.400, (1.517)	6.800, (1.304)	0.667
Non-Question Quality	6.286, (2.054)	7.000, (1.871)	7.000, (2.739)	1.000

Table 6.2: Results of the final user test on grammar, conversation flow and message quality on a 10-point Likert scale.

Three users got a different conversation flow where they were not given an opportunity to ask the agent questions. This happened when the chatbot was unable to extract the favorite sport of the user. The following quote and interaction was one of the three.

"It just isn't how a person would talk. I feel that it is very limited, all the things i can input are one single word. It feels shallow."

Shallow, single word interaction example

User: hi

Chatbot: SYSTEM: PLEASE enter your email address to continue.

User: REDACTED_EMAIL

Chatbot: Hi! What's your name?

User: REDACTED_NAME

Chatbot: Cool!

Chatbot: What's one sport that you like?

User: Sailing

Chatbot: My favorite sport is soccer.

Chatbot: Here's another question, what's your one favorite pet?

User: Dig

Chatbot: I see. My favorite animal is sheep.

Chatbot: Thank you REDACTED_NAME! You have now finished the first session, please go back to the form for a small survey, and start a second session in a few days. I will send you a reminder via email.

Feedback from three different users regarding conversation flow

"Sometimes the bot will just answer with a closed answer, so you can't really keep the conversation going and it won't start a new subject out of itself. Maybe if he ends with a closed answer he can bring up a new subject himself"

"The way the chatbot asked questions about things we talked about in our previous conversation felt a bit...wooden? I think the questions should be phrased in a more natural "hey we are texting with each other" kind of way."

"The chatbots answers ("Mh") were pretty unengaging, causing the flow of the conversation to come to a halt."

Extracted example illustrating "Mh"-acknowledgement

...

Chatbot: Last time we talked about Yoga, did you do it recently (<30 days)?

User: Not since we last talked, but before (<30 days) - yes.

Chatbot: Mh okay.

User: Did you play soccer?

Chatbot: No I played a little bit at the junior college I grew up in, it was very tough.

Chatbot: We also talked about Cat. Do you happen to own one?

User: I do, yes.

Chatbot: And what color?

User: She's tricolor - white, black, yellow.

Chatbot: Mh.

...

Feedback from two users regarding grammar

"there were sentences like "If I have to choose a favorite sport, I think I would have to choose soccer." ("If I have to choose an X, I choose x" or "If I had to choose X, I'd choose x")."

"Grammar was sometimes wrong, eg. "last time we talked about cat". Responses to my questions were also a bit off."

Extracted example illustrating grammar and response flaws

...

Chatbot: Good day! How are you?

User: Doing well, you?

Chatbot: I'm doing pretty well.

Chatbot: Last time we talked about Skiing, did you do it recently (<30 days)?

User: Hahaha no I didnt go skiing recently

Chatbot: Mh okay.

User: Last time I went was 2 years ago

Chatbot: No, this time I only went once.

Chatbot: We also talked about cat. Do you happen to own one?

User: We have 2 cats

Chatbot: Mh.

User: And you?

Chatbot: Nothing really.

...

Two full interactions (cherry-picked), from session 1 to session 2, are provided in the appendix A.9.

7 | Results - Chatbot Comparisons

Three different chatbots were developed where a user test was held for each chatbot. The user test on the question answering chatbot had 32 participants in total, where 16 of the participants were deemed as non-ideal subjects. The user test on the question asking chatbot had 5 participants. The final user test on the integrated chatbot had 17 participants in the first session out of which 3 were discarded. The second session had 10 participants out of which 5 were discarded. The results of the user tests performed on the three chatbots are shown in table 7.1. The table compares the chatbots on 1 session to measure the user's perception after interacting with the agent once. The results of the user tests from the integrated chatbot and question asking chatbot during both sessions are compared and shown in table 7.3. Table 7.2 shows P-values found from performing ANOVA tests on the measures of engagingness and realness. With a significance level of $\alpha = 0.05$, it is found that there is a significant difference in realness between the groups when including the Question Answering 'All' group in the test, but not when excluding it. Furthermore, there is no significant difference between the groups in table 7.3 as indicated by the p-values.

Group	Integrated Agent (Session 1 - All)	Question Asking (Session 1)	Question Answering (Ideal)	Question Answering (All)
Participants	14	5	16	32
Engagingness	3.571, (0.852)	3.200, (0.447)	3.688 , (1.138)	3.281, (1.301)
Realness	2.786, (1.122)	3.600 , (0.548)	2.375, (1.147)	1.969, (1.177)
Retention	3.429 , (0.756)	-	3.188, (1.515)	2.938, (1.458)
Persona	3.071, (1.072)	-	3.250 , (1.528)	2.750, (1.391)
A-Relevancy	3.714 , (1.490)	-	3.250, (1.125)	2.750, (1.136)
Q-Relevancy	3.857, (1.406)	4.400 , (0.548)	-	-
Grammar	4.143, (0.930)	4.400 , (0.548)	-	-

Table 7.1: Comparing survey results of the first session of different chatbots by looking at the mean values supplemented with the standard deviation.

	P-value	P-value Excl. Q-Ans (All)
Engagingness	0.623	0.618
Realness	0.012	0.098

Table 7.2: Resulting P-values of ANOVA tests on the chatbots' engagingness and realness scores. The third column (far-right) excludes the 'All' group of the Question Answering chatbot.

Additionally, box plots were created to display the distribution of the test results presented in table 7.1, based on a five number summary: minimum, first quartile (Q1), median, third quartile (Q3) and maximum. Figure 7.1 depicts the engagingness metrics from table 7.1. The integrated chatbot had an evenly spread result. The question asking chatbot did not have a large amount of participant, therefore resulting an uncompleted box figure. The question

	Integrated Agent (Session 1)	Integrated Agent (Session 2)	Question Asking (Session 1)	Question Asking (Session 2)	P-value
Participants	5	5	5	5	-
Engagingness	3.600 (0.548)	3.000 (0.707)	3.200 (0.447)	3.200 (0.837)	0.541
Realness	2.400 (0.548)	2.800 (0.837)	3.600 (0.548)	2.600 (1.140)	0.137
Q-Relevancy	4.400 (0.894)	4.200 (0.837)	4.400 (0.548)	3.800 (1.304)	0.716
Grammar	3.933 (1.023)	4.022 (0.659)	4.400 (0.548)	4.800 (0.447)	0.232

Table 7.3: Comparing results of shared measures between the integrated chatbot and the question asking chatbot, illustrating for each question the mean value supplemented with the standard deviation. Additionally, the resulting p-values of ANOVA tests are presented.

answering chatbot with ideal results had a wide spread of score, with a majority that found it to be engaging; whereas the question answering chatbot with all results had a wider spread and overall slightly lower score on the engagingness.

Figure 7.2 depicts the realness metrics from table 7.1. Most participants perceived that the integrated chatbot was not human-like except for two outliers. The question asking chatbot had a score that was above average on the 5-point Likert scale. The question answering chatbot, both with ideal results and all results, had the same spread of score (range). The median line illustrates that the 'ideal' group for the question answer chatbot had overall higher score compared to the 'all' group.

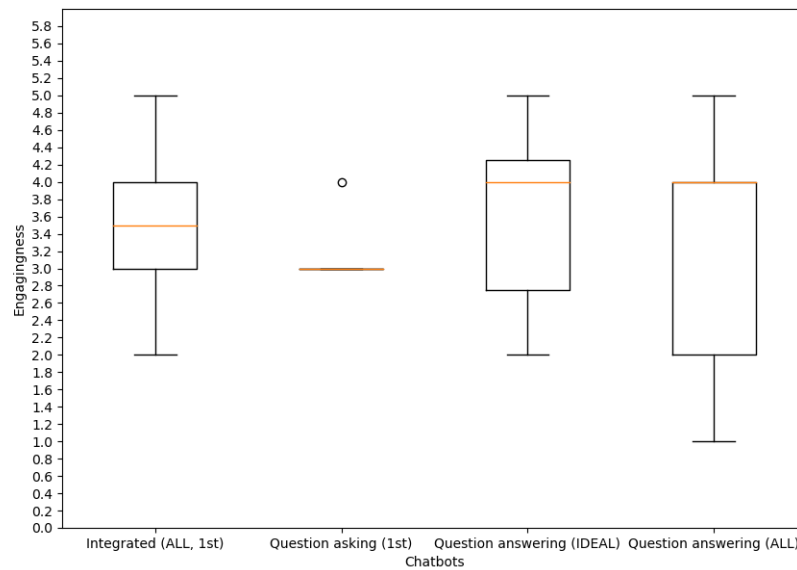


Figure 7.1: Box plot of the three chatbots' engagingness in the first session. The whiskers are at a distance of 1.5 interquartile range length (IQR).

To make any conclusions regarding the results of the user tests on how engaging the agents are, a comparison is made between other social chatbots. The comparison is made based on one session (first) due to the other agents being tested for one session/interaction.

The researchers who created the Persona-Chat dataset tested different models trained on the dataset, where they measured engagingness, fluency, consistency and persona detection. Fluency was measured from 1 - "not fluent at all" to 5 - "extremely fluent". Engagingness was measured from 1 - "not engaging at all" to 5 - "extremely engaging". Similarly, consis-

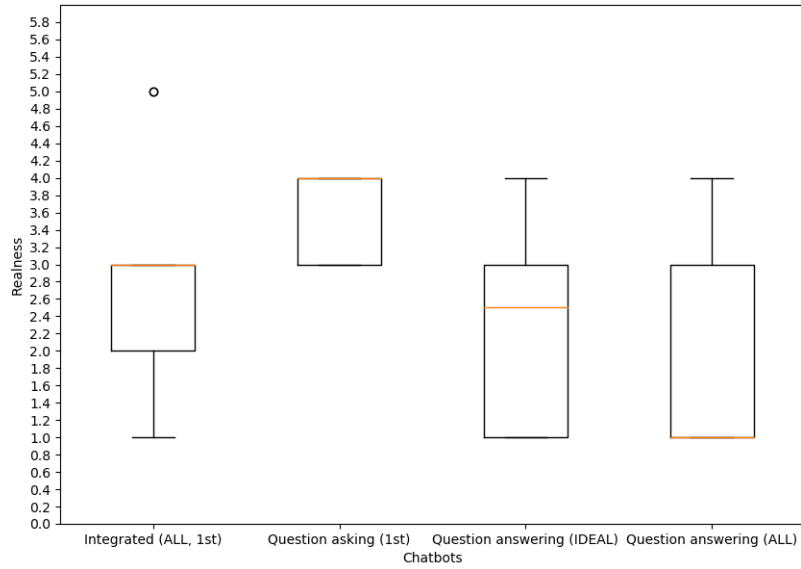


Figure 7.2: Box plot of the three chatbots’ realness in the first session. The whiskers are at a distance of 1.5 interquartile range length (IQR).

tency was measured from 1 to 5, provided the example “I have a dog” followed by “I have no pets” is not consistent. A comparison between their trained models and the agents in this thesis is illustrated in table 7.4. In the table, “profile memory” represents a model that takes into consideration a provided persona. Interestingly enough, the engagingness scores are lower in this case for the models that take persona into consideration. Persona detection was measured by presenting two alternatives and letting the human user say which persona they think they interacted with [56]. In this thesis, a persona is not considered in the same way, e.g. the generative model does not take into consideration a pre-defined persona when generating the next response. The persona measure is not compared due to it, in this case, being based on whether the user perceive that the agent has its own personal background and personality, rated on a 5-point Likert scale. In this thesis, inconsistency is measured instead of consistency. The results are (naively) reversed to represent consistency instead, e.g. inconsistency measure 2.75 reversed to consistency measure 3.25. The results from the surveys in this thesis measuring engagingness, “The chatbot is engaging/interesting to interact with.”, is used for comparison on engagingness. The measures of grammar are used for comparison in the fluency column in this case. The final test (Integrated Agent) measured grammar on a 1-10 scale and the result has therefore been scaled down to 1-5 for comparison. It is important to consider the difference in sample size when reading the table, as the Persona-Chat models used 100 dialogues for evaluation, compared to the varying sample size of 5-32 for the chatbots in this thesis.

After the release of the Persona-Chat dataset, a competition was held called ConvAI2, where models were trained on the task of acting out a persona. In table 7.5, a comparison of engagingness is made between the agents in this thesis and agents designed on the persona-chat task, as presented in the ConvAI2 challenge [12]. The agents in the challenge were evaluated through human evaluation where the dialogues were of length 4-6 turns each. The human evaluators were asked to answer the question “How much did you enjoy talking to this user?” on a scale of 1-4. In this thesis, scales were from were mainly from 1-5, and as such, normalized scaling was performed for the sake of comparison. It should be noted

Agent	Fluency	Engagingness	Consistency
Seq2Seq	3.17 (1.10)	3.18 (1.41)	2.98 (1.45)
Seq2Seq - Profile Memory	3.08 (1.40)	3.13 (1.39)	3.14 (1.26)
KV Memory	3.81 (1.14)	3.88 (0.98)	3.36 (1.37)
KV Profile Memory	3.97 (0.94)	3.50 (1.17)	3.44 (1.30)
Question Answering Agent (All)	-	3.28 (1.30)	2.81 (1.28)
Question Answering Agent (Ideal)	-	3.69 (1.14)	3.25 (1.18)
Question Asking Agent (Session 1)	4.40, (0.56)	3.20, (0.45)	-
Integrated Agent (Session 1 - All)	4.14, (0.93)	3.57 (0.85)	-
Human	4.31 (1.07)	4.25 (1.06)	4.36 (0.92)

Table 7.4: Comparing the agents in this thesis with models from the Persona-Chat paper [56] on fluency, engagingness and consistency.

Agent	Engagingness (1-5)	Engagingness (1-4)
Lost in Conversation	3.81	3.11
Hugging Face	3.24	2.68
Little Baby	2.92	2.44
Question Answering Agent (All)	3.28	-
Question Answering Agent (Ideal)	3.69	-
Question Asking Agent (Session 1)	3.20	-
Integrated Agent (Session 1 - All)	3.57	-
Human	4.31	3.48

Table 7.5: Comparing engagingness of the agents in this thesis with the top 3 agents presented in ConvAI2 as well as human for reference [12].

that the agents in this thesis differ from the ones developed for the Persona-Chat task, as those agents are able to act out a persona given a few lines of descriptive text. Again, it is important to note the sample size difference, as the Persona-Chat agents were evaluated on 100 dialogues each.

The tables show that the engagingness and consistency of the developed agents are similar to that of chatbots in the Persona-Chat task. However, compared to the human baseline (of humans who acted out a persona), the results of the chatbots are still not near human level engagingness.

8 | Discussion

This chapter consists of discussions revolving what can be observed from the results of the user tests, the result compared with previous works, potential improvements, and ideas for future work.

8.1 Results

This section discusses the results from the user tests of the two individual chatbots in this thesis, and how the results were affected as the chatbots were merged.

8.1.1 User Test - Question Answering Chatbot

The results of the first user test are illustrated in table 3.3. The test consisted of some ideal interactions and some non-ideal interactions, which led to the splitting of the user group into two groups "ideal" and "non-ideal" with 16 subjects each. Unsurprisingly, when comparing the two groups, the "ideal" group scored more positive on every measure whereas the "non-ideal" group scored less positive on every measure. T-tests showed that there were significant differences between the means of the two groups on most measures. It was also found that the ideal group had 31 out of the 41 non-generated responses which supports the claim that the non-ideal group consists of lower quality samples for the test, and should have less weight on the result as a whole. This is because non-generated responses (from template and retrieval components) are the core of what the question answering agent was designed to handle (personal question answering).

Engagingness & Retention

When answering whether the agent was interesting to interact with, the result was rather neutral for the "All" group. A hypothesis was that introducing artificial response delay would lead to higher engagingness compared to instantaneous responses. The hypothesis could not be accepted based on t-tests which showed that there was no significant difference between the groups in terms of engagingness. Additionally, the NO-DELAY group measured higher engagingness which could suggest the opposite; instantaneous responses are more engaging for users when interacting with a social chatbot.

Realness

The users did not feel as if they chatted with a real person. For this question, and the user survey in general, an optional text field would have been beneficial to explore potentials for improvements. One reason for the low perceived realness may have been due to the agent's inability to answer follow-up questions, which led to irrelevant responses that may have ruined the immersion. The score may also have been affected by the fact that the agent was only capable of answering questions, which does not simulate a real conversation which consists of both question asking and question answering. The second hypothesis was that the group with artificial response delay (DELAY group) would perceive the agent as more real or human-like than the group with instantaneous responses. While the results show a

significant difference ($p=0.037$) between the ideal DELAY and ideal NON-DELAY group, the opposite effect was observed; the ideal NO-DELAY group perceived the agent to be more human-like than the ideal DELAY group.

Responsiveness / Unresponsiveness

Overall, the users did not feel as if the response time (average 5.983 seconds) negatively affected the chatting experience. This knowledge can be utilized in future design of chatbots as it may allow for features, models or components with higher time complexity that may have otherwise been disregarded to achieve near instantaneous responses. The measure of unresponsiveness was the most significant difference ($p=0.009$) between the ideal and non-ideal groups. The ideal group scored very positively on this measure, meaning that they were fine with the response time. The non-ideal group scored more toward the neutral. Such outcome is expected as the non-ideal group had different expectations and interacted with the agent differently, and occasionally out of its intended application. As the agent responded with something irrelevant to their input, the response time may have been found irritating as the user had to wait all that time for an unintelligent reply, in relation to their input.

The NO-DELAY group scored better on responsiveness while the DELAY group scored better on unresponsiveness. The reason why the DELAY group scored better on unresponsiveness is probably due to the consistency of the response times. The artificial delay masked the calculation time of the generative model for this group, whereas the group with no delay had the possibility of experiencing instantaneous answers from the template and retrieval components. When this group then received an answer from the generative model, the answer was observably much slower than non-generated answers. This in turn may have led the user to feel as if the response time of the system was less acceptable.

Repetitiveness

While the NO-DELAY group scored higher on every measure except for unresponsiveness and repetitiveness, it is uncertain as to why the DELAY group perceived the agent to be less repetitive. Perhaps the delay made it seem as if the agent took the time to process the user's input and its next reply. When considering the results of the t-tests, the more likely answer is that the group scored higher by chance ($p=0.518$).

Inconsistency, Relevancy & Persona

There was a significant difference when comparing the ideal DELAY and ideal NO-DELAY groups on inconsistency ($p=0.038$) and relevancy ($p=0.005$) (furthermore $p=0.054$ on persona). After analyzing the conversations of the ideal group, the results show that the ideal NO-DELAY sub-group had fundamentally better interactions compared to the ideal DELAY sub-group. This was measured by comparing the number of non-generated responses, the percentage of responses that made sense to the input, and the number of follow-up questions asked. The ideal NO-DELAY group asked less follow-up questions, got more non-generated responses, and had a slightly higher percentage of responses that made sense. All of which factors may have contributed to why the group scored higher on most measures compared to the ideal DELAY group.

The results show that instantaneous responses result in higher perceived realness compared to artificially delayed responses. However, it should also be noted that analysis found that the ideal NO-DELAY group by chance had an overall better experience interacting with the chatbot than the ideal DELAY group, which in turn may have impacted the results of several measures. One finding was that the ideal DELAY group was less negatively affected by longer response times, for this specific case where one component (the generative component) takes substantially longer than the other components.

Further tests are necessary to come to any general conclusions due to the small sample size and due to each user having different interactions. It is important to consider the vari-

ability of the user test when looking at the statistical aspect as to how artificial delay affected engagement and perceived realness in this study. Furthermore, the average response time of the messages in the NO-DELAY group was not significantly different from that of the DELAY group (5.103 compared to 6.646) due to response time of the generative model. Therefore, it may be important in future tests that the NO-DELAY group experience consistent response times (each response near instantaneous) to determine the effects of artificial/dynamic response delay. Additionally, the artificial delay may have been too large to observe the positive effects found in the study by [14].

A lesson learned from the study was to limit the scope of the users' interaction with the agent to prevent large variance in the outcome. For example, by telling the user to ask questions in a limited number of supported topics like hobbies, sport and food.

8.1.2 User Test - Question Asking Chatbot

The result of the user test with the question asking chatbot is illustrated in table 5.2. From a total of five interactions, three were considered as normal conversations which correspond to what a real life conversation would be carried out; and two were considered as unrealistic, as they were also testing the limitation of the chatbot.

Through a preliminary assessment of the results, it was observed that some users did not acquire a clear understanding of what they were supposed to do. Thus, improving information clarity will be considered in the future works, as well as what the chatbot was able to handle in general. The user interface only stated that the users were invited to "interact with a question generation chatbot", it should have also mentioned rules such as "refrain from asking questions to the chatbot". When the users started to ask questions, which were not supposed to happen, and they did not receive an appropriate answer, the user experience is then lowered, therefore leading to a lower score. It was clear that the final survey for the integrated chatbot should have a clear instruction such that users have a rough idea on what to expect and what they should do, in order to minimize the numbers of unserious interactions. This will also help to gather information that are usable for answering the research questions.

Due to different users had different experiences with the chatbot, depending on if the conversations were realistic or not, different scores were given. The users that had a normal conversation, felt that the chatbot was fun and interesting to interact with, although there were some missing features such as capitalization of words or identifying a typo. The users that had a unrealistic conversation, felt that the chatbot did not really understand. However, it was still fun to interact with because how the chatbot replied normally when it was given an out-of-context answer, which happened mostly in the second interaction, likely because the users wanted to try new inputs and were curious to see what the chatbot could handle and what kind of results they would get. A decrease in the realness metric was expected as the chatbot could not answer such messages. It is debatable whether or not these users should be counted as outliers, because such conversations do not take place during everyday talk between two people. On the other hand, it is common for users to write random and out-of-context messages when interacting online.

8.1.3 Final User Test

By comparing Session 1 and Session 2 in table 6.1 and table 6.2, we find that realness, persona, user memory and question quality increase in the second session. The measure of perceived user memory increased with statistical significance as the chatbot asked long-term memory generated questions which in turn is believed to have increased the measure on realness, persona and question quality. Although the realness, persona and question quality measures increased with the use of long-term memory, engagingness and the conversation flow decreased. With a less natural conversation flow, it is understandable that the engagingness may decrease as a consequence. User feedback also mentioned faults in the conversation

flow. Additionally, the first session made use of the question answering API to get self-disclosure while the second session only called the API if the user asked questions, which may have affected the engagingness and conversation flow. The measure on question quality increased in the second session, which may be correlated to users finding a greater interest in the long-term memory generated questions.

A couple of unforeseen scenarios happened during the test such as the system not recognizing the user for a second session. As such, there should have been better instructions emphasizing the importance of using the same e-mail for the study. Additionally, some issues occurred due to time constraints where the development required more time for completion. This can be observed in faults in the conversation flow, where some non-question user utterances were incorrectly provided to the question answering API to respond to. Additional faults were related to the question answering component being unable to extract entities which resulted in a different conversation flow for users in the first session, and the agent asking about "fail" in the second session.

One aim with the test was to compare engagingness and realness of the question asking chatbot before and after the final test to find how the user's perception of the chatbot was affected by the chatbot's personal memory and self-disclosure ability. This question remains partially unanswered due to the method more so measuring the effect of adding more features to the chatbot, such as question answering. As such, the survey should have consisted of additional questions that ask more specific questions about the user's perception of the chatbot's self-disclosure. An example of more specific questions would be:

- "The chatbot's personal background contributes to engagingness."
- "The chatbot's self-disclosure makes the chatbot seem real."

Where the questions or statements are rated from "not at all" to "very much". More specific questions would in turn reduce the amount of speculation around the meaning of the results.

8.1.4 Comparisons of the Three Chatbots

Table 7.1 compares the results of the three user tests. Although engagingness is slightly higher in the "Question Answering (Ideal)" group compared to the integrated agent, the retention is higher in the integrated agent. By looking at the box plot of the chatbots' engagingness in figure 7.1, it can be observed that the integrated chatbot has lower variance on the measure compared to the question answering chatbot. Based on the greater median and the smaller lower quartile, it can be said that the question answering chatbot has a greater mix of engaging and non-engaging interactions than the integrated agent which instead has a mix of engaging and neutral interactions (engagingness \geq 3). A logical assumption could then be made that the increase in retention is correlated with the lower variance of engagingness and the increase of the lower quartile (higher lows). The lower variance may in turn be correlated with the reduced scope of the users' interactions between the two chatbots. In other words, users are more likely to have an acceptable interaction (neutral or better) with the integrated chatbot than the question answering chatbot and in turn be more likely to interact again in the future.

The integrated agent suffered a slight loss of persona compared to the question answering agent which may be due to the question asking component having the main control of the conversation flow, leading to less self-disclosure. Answer relevancy increased in the integrated agent which may be contributed to the improvements of the generative model.

By comparing the question asking agent to the integrated agent, the engagingness increased in the integrated agent, whereas realness, question relevancy and grammar decreased. The decrease may be related to how the question asking agent had full control of the conversation by always asking scripted questions, compared to the integrated agent which acknowledged the answer, shared its own answer (self-disclosure), and allowed for

the user to ask questions. Being able to interact with an agent more than simply answering its questions exposes the limitations in the chatbot's capabilities and intelligence. It can also be observed that a high realness score does not necessarily imply a more engaging chatbot. However, it is also important to note the low sample size of the user tests, especially for the question asking agent, which makes it difficult to make generalized conclusions.

8.2 Method

The method of the three agents are discussed individually in the following sub-sections. More research into the methodology could improve design and development of the question asking and answering chatbot.

8.2.1 Question Answering Agent

Retrieval and Generative components

The retrieval structure was created in a very manual fashion, with an ID that connects a question and a potential answer. The ID allowed for remembering which question types had been answered previously, such that the agent could retrieve the past answer and to avoid giving two different answers to the same question. This method has several downsides however:

1. Requires manual maintenance to extend the database by considering answer ID.
2. New questions that are different but with high cosine similarity to a past question are answered by the saved past answer.
3. Contradiction can still occur by answers from the generative component.

Suggested alternative method is to create a question-answer pair retrieval database. Given a question, retrieve N number of candidate responses, for example the top 10 questions that are most similar to the user's question. This method is used by XiaoIce [59] to some extent, where they have a filtered database to only retain responses that are aligned with the pre-defined persona of their chatbot. Using a question-answer pair retrieval database would allow for easier maintenance, as any question-answer pair can be added as long as the answer makes sense in relation to the question. Generated responses can then extend the database as long as there is an evaluation (manual or otherwise) to filter out bad pairs. Then by saving all questions and answers in conversations, new questions can be compared to previous questions, e.g. by cosine similarity, to determine whether the question has been answered in the past. If it has been answered in the past, then the agent can re-use the past answer to avoid contradicting itself. However, consider the question "What do you do for a living?" followed by "What is your job?". The similarity score may not find that the questions are similar enough to retrieve the past answer, and thus the agent may contradict itself, by first saying it is a receptionist and then a baker. A suggestion is to have memory slots, similar to what Mitsuku has for user modeling 2.5, but for the agent, where the agent's persona is built up as the user interacts with it. Then, a new ranking need to consider whether a response contradicts the persona or not. The ranking may use BERT or another Transformer model trained on Natural Language Inference, to detect contradiction between new candidate responses and past answers/persona. An example of memory slots would be name, age, work, hobbies, nationality, location, etc. Additional slots would be family members and friends, where the family members and friends have their own slots for name, age, etc. In general, more work could have been spent on the personality aspect, for example by creating template personas. With pre-defined personas it is possible to restrict the amount of contradictions by having persona specific retrieval databases. A measure would still be necessary to ensure that generated answers do not contradict the persona, however.

With the suggested question-answer pair database, it is possible to keep expanding the database from user interactions. The database can then be used to train the generative model. The trained model can then generate questions and answers which can then be added to the retrieval database. This cycle of expanding the database and training the model can continue as long as the bad question-answer pairs are filtered out. The downside is that manual filtering bad pairs by going through thousands of question and answer pairs is time consuming.

With the final version of the generative component, the component is fast enough to be used at all times. As such, whenever a question is not considered a template question, then instead, always retrieve a number of candidate responses from the retrieval database and generate some responses. Thereafter apply ranking to the retrieved and generated responses. Then use a threshold to filter out bad responses, and randomly select one of the remaining responses as the answer to output to the user. This method of response selection was used in XiaoIce, where it allows for variability. In comparison, the existing ID based retrieval structure in this thesis also returns a random response out of the answers with the ID matching the question.

As of now, the generative model may generate responses that contradict the sentiment memory. The responses may even contradict themselves, "I don't like to read but I do enjoy reading". This may be avoided by implementing Semantic Role Labeling to split the sentence into clauses and then by using VADER sentiment analysis on each individual clause. Semantic Role Labeling finds the semantic meaning of words or phrases in a sentence. By using sentiment analysis, we can find whether a response contradicts the sentiment memory, and if so either discard the answer or replace the keyword to fit with the sentiment memory. For example, if the generated answer says that the agent likes basketball whereas the sentiment memory has a negative sentiment for basketball but a positive sentiment for football, then basketball can be replaced by football before the answer is output to the user. Then, the existing templates could be completely replaced by the generative model, which increases variability and in turn should positively affect engagingness. Some sentences may include mixed sentiment about different subjects, such as "I don't like basketball, but I enjoy football". Then it may be possible to split the sentence into clauses or n-grams and run sentiment analysis on the individual clauses to find the corresponding sentiment for each subject, which in turn can find contradiction if there are mixed sentiments about the same subject in a sentence.

A possible mistake that was done during fine-tuning was that model #8, #9 #10 was only fine-tuned on a question answering dataset while model 7 was fine-tuned on a movie dialogue dataset first and then question answering. The mistake was stopping the behavior of first fine-tuning on the movie dialogues, as they may have contributed to more engaging generated text. A suggestion is therefore to first train on movie dialogue data or Persona-Chat data to first attain interesting conversations, and then fine-tune for question answering.

For the ideal length penalty in the ranking system, it may be beneficial to make it more dynamic, for example by trying to match the sentence length of the partner (user). This suggestion is based on findings in [36] where human users roughly matched the sentence length of their conversational partner. Additionally, some questions entail different answer lengths (e.g. yes/no questions), although this may be more difficult to consider and implement.

ConceptNet

ConceptNet's API was initially used to add unseen words to the sentiment memory by calling the API to find the IsA relationship (e.g. an apple is a fruit/food) and add the unseen word under the appropriate category. This feature was later disabled due to the increase in response time of the agent and replaced by a word embedding and similarity based method, which is less accurate. A solution would be to call the API in another thread such that the results can be used by the agent, not in the current response, but in future responses. As the sentiment memory only stores a noun if the IsA relationship corresponds with existing, pre-defined topics, an additional memory should be created to store every new noun. Then the agent does not need to call the API again and again for the same noun, e.g. consider the noun "name" when given the common question "What is your name?".

User test

The user test got mixed results as the users had different expectations of the agent and its capabilities. Even though the initial message by the agent stated "Feel free to ask anything about me, whether you want to know what food I like or what my hobbies are.", some users still asked task-oriented questions. To consider for future user tests is to give more specific instructions and to emphasize on what the agent can and cannot do. For example, by emphasizing on the fact that the agent is a question answering component and that it is only designed to answer personal questions. As well as telling the user to pretend like it is a person you are trying to get to know. Additionally, it might have been helpful to give examples of questions that the user can ask, e.g. "Suggestion: Ask a question about sport.". A concern when designing the user test was to not influence the user too much to allow for a better understanding of how a user may interact with the agent.

After the first user test, some time was spent on creating context memory for the generative model. Another solution, or a complement to the solution would be to transform user utterances into contextual utterances, as is done in XiaoIce [59].

What sports do you like?
I like soccer.
Why? -> Why do you like soccer?

Automatic evaluation

We have learned that automatic evaluation of response generation does not correlate well with human evaluation [25]. However, similar to [25], we believe that embeddings, such as the Universal Sentence Encoder, is a key component for automatic evaluation. The metric should aim to find topic relevance rather than word-overlap to some ground truth, as many different kinds of answers are still as relevant to a question, even if it is not similar to the ground truth.

A late realization for automatic evaluation when used in a one-versus-one setting such as comparing two methods or models, was that of using a Δ similarity threshold (the difference in ranking scores). By using a Δ similarity threshold it is possible to let the automatic evaluation decide if two answers are equally relevant (tie), as was done in manual evaluation. This might lead to slightly better correlation to the manual evaluation as answers with low difference in their similarity/ranking score would then be considered tied.

8.2.2 Question Asking Agent

The question asking agent switched topics very abruptly, due to its scripted approach. For future development it would be good to consider topic switching based on the flow of the conversation, for example when the user's or agent's inputs are getting bland such as "Ok" or "I see". Additionally, the agent would benefit in terms of engagingness by integrating with the 1-turn follow-up question generation by [27].

A question generation component could be made using the same GPT-2 model as seen in the question answering agent. The model is trained on question and answer pairs and is therefore able to both generate questions and generate answers. This requires a new ranking system that considers the conversation history to generate topic relevant questions. The ranker would also need to consider whether a question has been asked and answered by the user already to avoid repeated questions. Other than creating a ranker, there is complexity in how the history should be considered. The agent takes question and answer pairs as history input when generating but cannot see from the history whether a question was asked by the user or by the agent.

One other approach to tackle this issue is allow an idle time after the bot acknowledges user's previous message. When the timer reaches zero, and no additional input from the user was given, then the chatbot would continue to ask a question or initiate another topic. Such feature is available in Rasa, however, due to the nature of the framework requiring an intent

from the user in order for the chatbot to reply, some sort of activation intent is needed in order to trigger a certain action after countdown. Below is an example of the timer function, which calls an intent in order to trigger the chatbot's reply.

A feature in Rasa which allows scheduling of an intent to be triggered in a set amount of seconds

```
...
Chatbot: What's one sport that you like?
User: Nothing
(Timer elapses for 5 seconds)
User: \scheduled_event_trigger
Chatbot: Here's another question, what's your one favorite pet?
...
```

However, this approach was discarded due to the automated intent trigger that must be printed out as a user input. It was deemed that this trigger message would break the immersion, and thus decrease the user experience. It may be possible to hide this intent trigger message if the chatbot was deployed on a website or an app, where the developer has more control on the back-end and is able to decide what will be shown in the user interface.

8.2.3 Integration

The final agent is only able to consider one intent per message and may therefore not work ideally for a user input consisting of a statement and a question. Future work should aim toward using tools such as Semantic Role Labeling, to create clauses to identify multiple intents, and to potentially handle those intents as individual inputs.

8.3 The Work in a Wider Context

For the work to be further developed and used commercially, it is necessary to add additional filters, especially for the generative model in the question asking agent, to avoid inappropriate responses. As the GPT-2 model was trained on 40GB of internet text, it may be capable of generating vulgar slurs or insult the user. This is an important and shared issue between several pre-trained generative models [57] [1] [36]. Additionally, if the user asks what the agent thinks about a subject, the current template component may say it either likes or dislikes it. This may lead to inappropriate responses if the subject noun is e.g. racism, slavery, torture, etc. If additional measures are added, such as appropriate filters, then the work can be utilized in future chatbots. The question answering agent can be used as an API to extend existing chatbots, providing user specific conversation memory and user specific agent persona. It is hoped that the work done may be helpful for other developers or researchers toward the development of long-term engaging social chatbots. Especially it is hoped that the work done around the generative component, with the suggested ranking system for over-generated answers, and the context understanding based on conversation history, may serve useful for cost-efficient corpus-based approaches when building chatbots.

One major security improvement that can be done is the user identification. Currently the chatbot identifies the users using the email address that they have provided. It is a method for the chatbot to recognize the user and decide whether it is the user's first-time interaction or not. This simple solution works in the current scope since the testers are mostly either students or colleagues from the university. However, there is no implementation on verifying the email address, such as sending a confirmation link; therefore, it is possible to enter a false one. Additionally, there is no protection such as password, if a user knows someone's email address, it is possible to fake an identity. As the current progress, it will not be a huge concern since the numbers of topics are narrowed down to two, and the information given by the users are not extremely personal.

8.4 Future Work

It would be interesting for future work to consider a component, either by training a generative model or by using rules to generate daily "stories", as inspired by [15]. For example, by training a GPT-2 model to generate sentences of the sort "Yesterday I went to the beach and played volleyball.", "Tomorrow I want to go to the park and feed ducks.". The generated stories should consider the persona of the agent and should allow for the user to ask follow-up questions. The main time-consuming task for this component is the gathering or creation of the dataset, which is why it was not considered in this thesis due to time constraints.

Replika had an interesting feature where it allowed users to react on each message by the agent with a thumbs up or thumbs down symbol. They then trained a BERT classifier with the help of the feedback from the users. The classifier was then used to rank candidate responses based on if it was predicted to get a thumbs up or thumbs down. This would be a good feature to use and implement in ranking, but it requires having an existing active user base that makes use of the feature for a longer period of time to collect the data.

It would be interesting to investigate the concept of empathy vectors as seen in XiaoIce and Replika, together with training a model using the Persona-Chat dataset to achieve persona specific generated responses.

In order for the bot to be used commercially, some security measures should be taken, such as user verification, password, two-step verification using a secondary email address or phone number. The email given by the user during registration should also be verified.

Recent work

New models have been researched and released since the start of this thesis. On June 11th, 2020, OpenAI revealed an API around a new model: GPT-3¹. The API is, at the time of writing, in a closed beta, and is intended for commercial use. The largest model in the GPT-3 series is 175 billion parameters, and the training data is over 570GB of text [5]. Among the API beta testers is Replika, where they have observed an increase in their users' happiness ratings by 20 percentage points². The API may be worthy to consider for future commercial chatbots or other text generative applications. Instead of fine-tuning, the model can be taught a task given a natural language description, a few examples, and a prompt. Transformers suffer from repetition, which was observed to still be the case for GPT-3. Additionally, generated text still tends to lose coherence over long passages and contradict themselves [5].

It is apparent that the Transformer models are growing larger and larger to the point where some researchers rather throw more data and computational power to achieve better result than to come up with new innovative architectures. This in turn motivates the need for optimization. One method, as discussed earlier, was to create distilled versions (2.2.4.2) that are smaller, faster, and still retain much of its performance [37] [42]. In this case however, the Reformer [21] was introduced as an upgraded transformer with lower computation complexity.

ERNIE-GEN [52] is a new model that achieved state of the art results on various Natural Language Generation tasks, such as Persona-Chat. This motivates further investigation regarding its capabilities as part of chatbot systems.

Additional research has been carried out revolving the Persona-Chat task, such as the development of a receiver-transmitter architecture [26]. Different from the competitors in the ConvAI2 challenge, this architecture takes into consideration the persona of the user in addition to its own persona, which allows it to ask personal questions. Based on this concept, question generation algorithm used in the current work is developed.

DialoGPT [57] used the GPT-2 model but trained extensively on conversational data. The model is basically a pre-trained chatbot by itself. The model is available in Hugging Face's library. From brief interactions with the existing pre-trained model, it seems to generate short

¹<https://openai.com/blog/openai-api/>

²<https://beta.openai.com/?app=creative-gen>

and not so engaging responses. It would have been interesting to fine-tune a DialoGPT model with the datasets created in this thesis and compare the results. In theory, the results should be better as the model is trained on millions of conversations instead of many different types of texts.

Researchers at Facebook AI created a "state-of-the-art open-source chatbot" called BlenderBot which was released end of April where it claimed to outperform other chatbots in terms of engagingness³. When compared one-to-one to Google's chatbot Meena [1], human evaluation found that users more often preferred BlenderBot. Once again, a transformer architecture was chosen where different sized models were trained, smallest one having 90 million parameters and the largest having 9.4 billion parameters. The models are trained to be able to display knowledge, personality (Persona-Chat dataset) and empathy, and to blend these skills seemingly. Had the model been released earlier, it would have been tested and considered for the project as the main chatbot, with more focus spent on introducing the long-term memory aspect and reducing contradictions. Although the chatbot is observed to be engaging in few turn dialogue settings, the research paper describes the flaws of the model [36]. The author(s) mentions that the bot would be repetitive and dull if users were to interact with the chatbot over the course of several days or weeks. This is due to the chatbot being incapable of remembering earlier conversations as it has a hard limit for the history that it considers. This supports the need for further work and research in chatbots with memory, especially how they can be added to generative models, more than just context-based understanding. It also illustrates contradictions or inconsistencies, which is common-place for generative model chatbots. Similarly to what was used in the ranking in this thesis, researchers at Facebook AI [36] found that encouraging longer generations helps with reducing, but does not solve the problem of bland/"safe" response generation, e.g. "I don't know", which is another problem with generative models. They also found that the model tends to generate common words too frequently and rare words too infrequently, compared to human distribution (which leads to a less engaging chatbot). This further supports the decision to reward rare words during ranking (4.4.4). It should be noted however, the underlying problem is the nature of the models themselves, as they try to find the most likely next word when generating output. Therefore, any ranking is limited by the underlying architecture. But it is possible to adjust parameters that affect the text generation such that the model chooses a "less likely" next word to achieve more engaging responses (which in turn may lead to less sensible sentences). As such, over-generating responses and applying length based and rare word based ranking should serve meaningful until the architecture is improved.

³<https://ai.facebook.com/blog/state-of-the-art-open-source-chatbot>

9 | Conclusion

Conversational agents have their prospective applications in society, whether it is to combat loneliness or to promote exercise. One of several important aspects for these agents to stay engaging is the use of memory to remember the user, retain a consistent personality and to avoid repetition.

The research question, *how can a personal social chatbot be developed with long and short-term memory, such that the interaction with the chatbot remains engaging over a longer period of time*, is investigated in five sub-questions, divided into development and user tests & evaluation.

1. *How can the user and agent specific memories be developed to extract and store information from user utterances, as well as utilizing the stored information in agent utterances?*

Information can be extracted using Natural Language Processing techniques, such as Part-of-Speech tagging to extract nouns and Named Entity Recognition to extract names. Neural models can also be trained to extract desired information by providing examples of user utterances (input) and the desired information (output). Sentiment analysis (e.g. with VADER) can then be used to find whether the user like/dislike or feel neutral towards a subject. The question answering agent's memory was developed to have a sentiment memory with subject-sentiment pairs for a number of topics. Templates were then used to utilize information, either from the agent memory, user memory or extracted from the user's most recent input.

2. *How can short-term memory be developed such that the agent can understand and provide a relevant answer to user utterances, such as follow-up questions, from the context of the conversation history?*

This question is answered from the research/related work presented in chapter 2. Three methods were observed:

1. Have memory slots for he/she/it, etc., as used in Mitsuku (2.4.2), where information is extracted from user utterances and stored in memory.
2. Transform user utterances into contextual utterances, as seen in XiaoIce [59].
3. When using neural/generative models, either design the architecture or fine-tune to explicitly consider history when generating new responses.

This thesis made use of the third method where a GPT-2 model was fine-tuned for question answering on conversational data where history was considered. The resulting model illustrates the ability to answer context dependent follow-up questions. It should be noted that the model also tends to generate nonsensical answers occasionally. Furthermore, recent advances in end-to-end trained neural conversational models [1] [36] present chatbots that converse well with respect to context understanding with the help of their short-term memory, but the chatbots

do not consider long-term memory.

3. *How is a user's experience with an agent affected when the agent generates questions which are personalized based on the user's utterances from a previous interaction?*

The users were able to realize when the chatbot asked questions that were related to previously discussed topics. However, for the question asking chatbot, the experience does not improve in terms of the perception of the chatbot's realness. Other metrics such as engagingness and appropriateness have similar scores during both interaction sessions. For the integrated chatbot, the engagingness decreased while measures of realness and persona increased between sessions. The hypothesis was that the users would find a chatbot more engaging and fun to interact with if it could remember information about the users. The hypothesis cannot be accepted due to the results of t-tests where no significant difference was observed between sessions. Furthermore, the hypothesis cannot be rejected because the users had personal expectations about the memory module which resulted in inconclusive outcome. Moreover, the results indicated that engagingness was independent of chatbot memory.

4. *How is the user's perception of the agent affected by the agent having a personal background, which allows for self-disclosure responses?*

The aim was to answer the question by comparing the engagingness and realness of the question asking chatbot before and after the integration with the question answering chatbot. The integrated chatbot resulted in higher engagingness compared to the question asking chatbot. Although the engagingness increased, the cause may also be due to the addition of more features, such as question answering, rather than it being due to the agent's personal background. The perceived realness was lower for the first session of the integrated chatbot compared to the question asking chatbot, but higher in the second session. The hypothesis was that the realness score would increase from the first to second session, as the chatbot would ask long-term memory generated questions. The hypothesis was not supported for the question asking chatbot as the realness decreased between sessions, while the integrated chatbot supports the hypothesis. Furthermore, no significant difference was observed on the measure of realness between the two chatbots and their respective sessions, indicating that the null hypothesis cannot be rejected. The results show that the question asking chatbot may initially give off a higher sense of realness compared to the question answering or integrated chatbot but leads to loss of realness after additional interactions.

5. *To what extent would the user's perception of the agent be affected if the user could ask the agent self-disclosure questions?*

By comparing the engagingness measure of the question answering chatbot and the question asking chatbot, the engagingness is higher for the question answering chatbot. Once the two chatbots were integrated, the engagingness was measured to be higher than the question asking chatbot by itself, while being slightly lower than the question answering chatbot. Therefore, giving the users the option to ask self-disclosure questions had a positive effect in terms of engagingness. The integrated chatbot had a lower perceived realness compared to the question asking chatbot, but higher compared to the question answering chatbot. Allowing users to ask the agent questions may lead to lower perceived realness due to the possibility of nonsensical and inconsistent responses.

Progress has been made in neural approaches for training end-to-end chatbots with context understanding but the methods do not consider long-term memory to remain engaging over longer periods of time. This thesis presents how generative-retrieval hybrid chatbots can make use of short and long-term memory to enable rapport building toward maintaining long-term engagingness by personalizing question and answer generation. User tests show that chatbots perceived as more human-like are not necessarily more engaging. The results of the user tests show that the developed chatbots achieved similar score on engagingness and consistency to that of chatbots in the Persona-Chat task (ConvAI2). However, the chatbots are still not near human level performance on these measures. Furthermore, it is observed that generative models are essential for open-domain conversation, but still suffer from inconsistency and unengaging or nonsensical responses.

Concluding example of the generative model

HUMAN: Do you think your answers make any sense?

MACHINE: This is a very difficult question to answer, I think you will find some that make sense.

Bibliography

- [1] Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. “Towards a Human-like Open-Domain Chatbot”. In: *arXiv preprint arXiv:2001.09977* (2020).
- [2] Christoph Bartneck, Dana Kulić, Elizabeth Croft, and Susana Zoghbi. “Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots”. In: *International journal of social robotics* 1.1 (2009), pp. 71–81.
- [3] Timothy Bickmore, Daniel Schulman, and Langxuan Yin. “Maintaining engagement in long-term interventions with relational agents”. In: *Applied Artificial Intelligence* 24.6 (2010), pp. 648–666.
- [4] David M Blei, Andrew Y Ng, and Michael I Jordan. “Latent dirichlet allocation”. In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.
- [5] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. “Language Models are Few-Shot Learners”. In: *arXiv preprint arXiv:2005.14165* (2020).
- [6] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. “Universal Sentence Encoder for English”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2018, pp. 169–174.
- [7] Yllias Chali and Sadid A Hasan. “Towards topic-to-question generation”. In: *Computational Linguistics* 41.1 (2015), pp. 1–20.
- [8] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches”. In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. 2014, pp. 103–111.
- [9] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *NIPS 2014 Workshop on Deep Learning, December 2014*. 2014.
- [10] Cristian Danescu-Niculescu-Mizil and Lillian Lee. “Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs.” In: *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*. 2011.
- [11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4171–4186.

- [12] Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. "The Second Conversational Intelligence Challenge (ConvAI2)". In: *The NeurIPS'18 Competition*. Springer, 2020, pp. 187–208.
- [13] Denis Fedorenko, Nikita Smetanin, and Artem Rodichev. "Avoiding echo-responses in a retrieval-based conversation system". In: *Conference on Artificial Intelligence and Natural Language*. Springer. 2018, pp. 91–97.
- [14] Ulrich Gnewuch, Stefan Morana, Marc Adam, and Alexander Maedche. "Faster is not always better: understanding the effect of dynamic response delays in human-chatbot interaction". In: (2018).
- [15] Rachel Gockley, Allison Bruce, Jodi Forlizzi, Marek Michalowski, Anne Mundell, Stephanie Rosenthal, Brennan Sellner, Reid Simmons, Kevin Snipes, Alan C Schultz, et al. "Designing robots for long-term social interaction". In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2005, pp. 1338–1343.
- [16] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [17] Kate S Hone and Robert Graham. "Towards a tool for the subjective assessment of speech system interfaces (SASSI)". In: *Natural Language Engineering* 6.3-4 (2000), pp. 287–303.
- [18] Karen Huang, Michael Yeomans, Alison Wood Brooks, Julia Minson, and Francesca Gino. "It doesn't hurt to ask: Question-asking increases liking." In: *Journal of personality and social psychology* 113.3 (2017), p. 430.
- [19] Clayton J Hutto and Eric Gilbert. "Vader: A parsimonious rule-based model for sentiment analysis of social media text". In: *Eighth international AAAI conference on weblogs and social media*. 2014.
- [20] Dan Jurafsky and James H. Martin. *Speech and Language Processing*. 2019.
- [21] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The Efficient Transformer". In: *arXiv preprint arXiv:2001.04451* (2020).
- [22] Iryna Kulatska. "ArgueBot: Enabling debates through a hybrid retrieval-generation-based chatbot". MA thesis. University of Twente, 2019.
- [23] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. "A Diversity-Promoting Objective Function for Neural Conversation Models". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016, pp. 110–119.
- [24] Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. "A Persona-Based Neural Conversation Model". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 994–1003.
- [25] Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. "How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 2122–2132.
- [26] Qian Liu, Yihong Chen, Bei Chen, Jian-Guang Lou, Zixuan Chen, Bin Zhou, and Dongmei Zhang. "You Impress Me: Dialogue Generation via Mutual Persona Perception". In: *arXiv preprint arXiv:2004.05388* (2020).
- [27] Yani Mandasari. "Follow-up Question Generation". MA thesis. University of Twente, 2019.

- [28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.
- [29] Sinno Jialin Pan and Qiang Yang. "A survey on transfer learning". In: *IEEE Transactions on knowledge and data engineering* 22.10 (2009), pp. 1345–1359.
- [30] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318.
- [31] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.
- [32] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language models are unsupervised multitask learners". In: *OpenAI Blog* 1.8 (2019), p. 9.
- [33] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *arXiv preprint arXiv:1910.10683* (2019).
- [34] Filipe N Ribeiro, Matheus Araújo, Pollyanna Gonçalves, Marcos André Gonçalves, and Fabrício Benevenuto. "Sentibench-a benchmark comparison of state-of-the-practice sentiment analysis methods". In: *EPJ Data Science* 5.1 (2016), pp. 1–29.
- [35] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.
- [36] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric M Smith, et al. "Recipes for building an open-domain chatbot". In: *arXiv preprint arXiv:2004.13637* (2020).
- [37] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108* (2019).
- [38] David R So, Chen Liang, and Quoc V Le. "The Evolved Transformer". In: *arXiv preprint arXiv:1901.11117* (2019).
- [39] Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, and Jasmine Wang. "Release Strategies and the Social Impacts of Language Models". In: *arXiv preprint arXiv:1908.09203* (2019).
- [40] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. "A hierarchical recurrent encoder-decoder for generative context-aware query suggestion". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 2015, pp. 553–562.
- [41] Robyn Speer, Joshua Chin, and Catherine Havasi. *ConceptNet 5.5: An Open Multilingual Graph of General Knowledge*. 2017. URL: <http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14972>.
- [42] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. "MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices". In: *arXiv preprint arXiv:2004.02984* (2020).
- [43] Ilya Sutskever, James Martens, and Geoffrey E Hinton. "Generating text with recurrent neural networks". In: *ICML*. 2011.
- [44] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

- [45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.
- [46] Oriol Vinyals and Quoc Le. "A neural conversational model". In: *Proceedings of ICML Deep Learning Workshop*. 2015.
- [47] Marilyn A Walker, Diane J Litman, Candace A Kamm, and Alicia Abella. "Evaluating spoken dialogue agents with PARADISE: Two case studies". In: *Computer speech and language* 12.4 (1998), pp. 317–348.
- [48] Richard S Wallace. "The anatomy of ALICE". In: *Parsing the Turing Test*. Springer, 2009, pp. 181–210.
- [49] Joseph Weizenbaum. "ELIZA—a computer program for the study of natural language communication between man and machine". In: *Communications of the ACM* 9.1 (1966), pp. 36–45.
- [50] Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. "Dialogue Natural Language Inference". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 3731–3741.
- [51] Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. "Transfer-Transfo: A Transfer Learning Approach for Neural Network Based Conversational Agents". In: *arXiv preprint arXiv:1901.08149* (2019).
- [52] Dongling Xiao, Han Zhang, Yukun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. "ERNIE-GEN: An Enhanced Multi-Flow Pre-training and Fine-tuning Framework for Natural Language Generation". In: *arXiv preprint arXiv:2001.11314* (2020).
- [53] Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. "aNMM: Ranking short answer texts with attention-based neural matching model". In: *Proceedings of the 25th ACM international on conference on information and knowledge management*. 2016, pp. 287–296.
- [54] Liu Yang, Junjie Hu, Minghui Qiu, Chen Qu, Jianfeng Gao, W Bruce Croft, Xiaodong Liu, Yelong Shen, and Jingjing Liu. "A hybrid retrieval-generation neural conversation model". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019, pp. 1341–1350.
- [55] Jianhua Yin and Jianyong Wang. "A dirichlet multinomial mixture model-based approach for short text clustering". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 233–242.
- [56] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. "Personalizing Dialogue Agents: I have a dog, do you have pets too?" In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 2204–2213.
- [57] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. "DialogPT: Large-Scale Generative Pre-training for Conversational Response Generation". In: *arXiv preprint arXiv:1911.00536* (2019).
- [58] Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. "Emotional chatting machine: Emotional conversation generation with internal and external memory". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.
- [59] Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. "The Design and Implementation of XiaoIce, an Empathetic Social Chatbot". In: *Computational Linguistics* 46.1 (2020), pp. 53–93.

A | Appendix

A.1 GPT-2 Text Generation

Cherry-picked example of GPT-2's text generative capabilities, with the prompt in italic.

The time of man was about to come to an end, and the time of the world was about to begin. And that time was not far off.

The great cities of the world were in a state of decay and decay was in the air. The great temples were crumbling to the ground, the great palaces were in ruins, the great farms were in ruins. And all the nations of the world were in a state of decay and decay. And the world was in a state of decay and decay. And in the midst of this decay and decay the voice of God was calling to men, saying,

"Behold, the days are coming, says the Lord, when I will make a new heaven and a new earth, for the former things have passed away."

And the voice of God said, "I am the God of Abraham, the God of Isaac, and the God of Jacob." And the voice of God said, "I will put my law in their inward parts, and write it in their hearts." And the voice of God said, "I will not leave you orphans, but will come to you, and will heal your land." And the voice of God said, "I will send the Spirit of my Father upon you, and will be your God, and you shall be my people. And the voice of God said, "I will be with you, and will keep you, and will bring you to your own land."

A.2 Fine-Tuned Generative Models and Their Answers

The following illustrates three different personal questions answered by four different models, where the models over-generated 8 answers.

What do you like to do in your free time?

Model 4

I like to drive.

Beer.

Well, the money was good for me to go to college.

I'm an electrician.

Sorta nothing.

There's always new things, silly.

I'm a dancer.

Swim and kayak.

Model 7

I spend time with my family, though I do enjoy traveling.

I spend most of my time turking.
I like to play video games.
I usually work out, but if I find something fun to do, then I will do it regardless of whether or not I get to do it in person.
I used to be a rock climber so I play rock climbing in my spare time.
I like to play video games.
I spend time with my family, go to festivals, go rock climbing.
I like to play video games.

Model 10

I like to watch movies, travel, go surfing, play video games.
Well, I enjoy golf.
I like to watch movies, travel, go surfing, play video games.
I like to watch movies, travel, go surfing, play video games.
I like to gamble, exercise, and go out to bars occasionally.
I like to watch movies, travel, go surfing, play video games.
I like to watch movies, travel, go surfing, play video games.
I like to watch movies and play video games.

Model 18

Well, I like to dance, read, go for hikes, and listen to music.
Haha nothing too free-minded.
I enjoy hiking and horseback riding.
I like to travel.
I spend most of my time exercising, especially track and tennis.
Well, like anything I will miss the porch in the morning, which is a shame.
I like to play video games, listen to music, and chat.
I watch, cook, take my dogs along, hike, go out on the weekends, watch a few movies, do some gaming.

Do you have any pets?

Model 4

Yes.
Yes, Mr. Milch-born and KAPOne.
No.
Gross.
No, thank you.
No, sir.
No.
No, I don't have any pets.

Model 7

I have 2 cats.
No pets ., never really considered that.
I'm not sure.
No pets of my own but my husband has 2 cats., I have 2 dogs who are both like my children, large and small.
I have 3 dogs.. they are 8 and 7.. no I don't have pets.
yeah, I have 3 dogs.
I have 3 dogs.. they are very house trained and love everything. They are also very healthy., Every little bit helps., They are 3½ years old and eating nothing but

carpet sections from their school
Wow, that's a lot.

Model 10

Yeah i have 2 dogs
No pets .
No pets .
Yeah, I have 3 dogs, a rat and a shekel.
Yeah i have 3 dogs.
Yeah, 2 guinea pigs.
Yeah i have 2 cats
Yeah i have 2 dogs.

Model 18

I have a dog.
No, but I do have a dog.
I have 3 dogs.
I have 2 dogs that I have with me whenever I have time.
I have 2 cats.
No, not currently.
Nope, no.
i've a dog as well.

What are your hobbies?

Model 4

To be a minister or a crusader or a magician, or just to have a talk show.
I'm into historical sports, and maybe I'm gonna write something about Dolly Parton.
Tennis, surfing, wrestling, surf... golf.
I'm a comedian.
I'm a computer programmer, a novelist, I'm a poet, and now I'm a doctor.
Sleep.
Spirits.
Fernie, I don't care about anything but her.

Model 7

that's awesome.
that would be traveling all day.
That sounds pretty awesome.
I like to play video games.
I like to ride my bicycle, and I like to play video games.
that sounds pretty good.
I like to play soccer a lot.
that would be awesome.

Model 10

My hobbies include biking, reading, playing video games, cooking, getting out and playing basketball.
My hobbies are golf, grand kids, the beach, cards and goofing off.
My hobbies are golf, grand kids, the beach, cards and hugs.

I like to cook, if you can call that a hobby and I like all types of craftwork
 My hobbies would be Netflix and yoga.
 I like to read, if you can call that a hobby but I also enjoy playing video games as well.
 My hobbies are golf, grand kids, the beach, cards and goofing off.
 Well I like to ride my bicycle around the area, play video games, play Magic: The Gathering, listen to music, cook, and probably get pretty cold there too.

Model 18

Well I work a lot, I play music mostly, I watch movies.
 Well, hobbies include hiking and boxing (which I'm really good at).
 My hobbies are golf and visiting friends in the area.
 Well I play tennis and I take walks along the river.
 Well, besides golf, I like basketball, crocheting, and crocheting on the beach.
 My hobbies are golf, restaurants and craft beers.
 So far so good.
 I enjoy chess, collecting cards, and spending time with friends.

A.3 GPT-2 Models Experiment

A short experiment was conducted to compare the different pre-trained GPT-2 models (124M, 355M, 774M, 1558M) on training time and text generation time. The experiment was conducted on the 17th of April, 2020 in a Google Colaboratory notebook by setting the runtime type from GPU to TPU, and by requesting more RAM (35.35GB). The models were fine-tuned and used to generate text by using the python library gpt-2-simple. The models were each trained 100 "steps" or iterations on the same dataset, which consisted of 28MB of dialogue extracted from the Persona-Chat dataset. The results are illustrated in table A.1 and table A.2. It is unknown as to why the 124M parameter model took almost as long to fine-tune as the 774M parameter model, in this case. The estimated training time should be 300-400 s, by comparing the training time growth between 355M to 774M, and 774M to 1558M.

Model	Elapsed training time, 100 steps
1588M	2689.62 s
774M	1424.34 s
355M	702.16 s
124M	1392.30 s

Table A.1: Comparison of training time between different GPT-2 models

Model	Average elapsed time to generate text
1588M	81.649 s
774M	44.203 s
355M	26.075 s
124M	11.331 s

Table A.2: Comparison of generation time between different GPT-2 models

To test the time it takes for the models to generate text, the pre-trained versions of the models (as released by OpenAI) were loaded without any fine-tuning and used to generate text with gpt-2-simple's function generate(). Each model called the function three times, and an average was thereafter calculated of the runtime. Additionally it was observed that setting a lower value on the length parameter in the generate() function, the time to generate was

reduced. In the case for the 355M model, a length of 40 took around 16 seconds compared to 23-26 seconds when length was 100. The 124M model then took 8-9 seconds compared to 10-11 seconds. A conclusion of the experiments was that, henceforth, only 124M parameter models are fine-tuned for the sake of having the lowest response time.

Parameters used in the generate() function

prefix="The secret of life is"

length=100

temperature=0.7

top_p=0.9

nsamples=5

batch_size=5

A.4 GTKY - Most Common Nouns

Occurrences	Noun	Example question
123	Time	What do you like to do in your free/spare time?
106+12	Kind/Kinds	What kind of <noun>do you like? (movies/books/food/music/sports/...)
98	School	Do you go to school?
96	Student	Are you a student?
90	Today	How are you today?
80	Day	how was your day?
73	Fun	What do you do for fun?
70	Weather	How's the weather?
67	Name	What is your name?
66	Work	what do you do for work?
64	Hobbies	What are your hobbies?
56	Things	what things do you enjoy doing?
55	Sports	Do you play any sports?
53	Music	What type of music do you like?
51	Job	What type of job do you do?
49	Area	What area are you from?
48	Part	what part of the country are you from?
47	Plans	What are your plans for the day?
41	Weekend	Do you have any plans for the weekend?
39	College	Are you in college?
39	Lot	Do you go to a lot of movies?
38	Year	What year are you?
37	Family	Do you have a family?
37	Movies	what type of movies do you like?
36	Kids	do you have any kids?
34	Food	what is your favorite food?
30	Type	What type of <blank>?
29	Summer	what are you up to this summer?
29	Country	What country are you from?
28	Place	do you have a favorite place you traveled to?
28	Life	Have you live in PA your entire life?
26	Games	What kind of games do you play?
25	Friends	what do you like to do with your friends?
25	Thing	what is your favorite thing to do?
25	Living	What do you do for a living?
24	State	What state are you from?
23	Studies	Do you like to do studies?
23	Fan	Are you a <sports>fan?
23	Years	How many years of Latin?
23	Snow	Did you guys get much snow?
22	Money	how much money do you usually make a day?
22	Interests	what are some of your interests?
22	mturk	How long have you been on mturk?
21	places	Do you know any good Indian places?
21	pets	do you have any pets?
21	stuff	what kind of outdoors stuff do you do?
21	children	Do you have any children?
20	City	what city do you live in?
19	Major	What's your major?
19+10	Books/Book	What kind of books do you like to read?
19	Field	What field are you in?
19	Way	Is there a better way?
17	Winter	Do you like the winter?
17	Movie	What's your favorite movie?
17	Person	are you an outdoorsy person?
16	Home	Do you miss home?
16	Study	What do you study?
16	People	where do you find these people?
16	Business	What kind of business?
16	Shows	What are your favorite tv shows?
15	Team	What's your favorite team?
15	Research	What kind of research are you doing?
15	Morning	how are you this morning?
15	Forums	are you in any forums?
14	Program	What is the program called?
14	Week	What do you usually do during the week?
13	Sort	What sort of work do you do?
13	Campus	What do you do on campus?
13	Guys	Did you guys get much snow?
12	Degree	what is your degree in?
12	Tv	what is your favorite tv show?
12+10	Dog/Dogs	what kind of dog?
11	Show	Whats your favorite show?
11	Basketball	Do you watch basketball at all?
11	Beach	Do you like the beach?
11	Football	Do you like football?
10	Masters	Do you have your masters?
10	Career	What are your career interests?
10	Undergrad	Are you an undergrad?
10	History	What interests you about history?
10	activities	What activities are you interested in?
10	instrument	Do you play an instrument?
10	World	where in the world do you live?
10	Season	What's your favorite season?

Table A.3: Top 87 (84+3) most frequently occurring nouns in extracted questions from the GTKY dataset.

A.5 Templates

question	answer_id	default_positive
Do you <sentiment><noun>?	1	0
What kind of <topic>do you <sentiment>?	2	0
What type of <topic>are you into?	2	1
What type of <topic>do you <sentiment>?	2	0
What is your favorite kind of <topic>?	3	1
What is your favorite <topic>?	3	1
What's your favorite <topic>genre?	4	1
What is your favorite type of <topic>?	3	1
What is your favorite kind of <topic>?	3	1
What do you think about <noun>?	5	0
What sort of <topic>do you <sentiment>?	2	0
What <topic>do you <sentiment>?	2	0
What <topic>are you into right now?	2	1
What are your favorite <topic>?	6	1
What are some of your favorite <topic>?	6	1
Do you have any interest in <topic>?	2	1
Do you have a favorite <topic>?	3	1
Do you <sentiment><noun>at all?	1	0
Do you have any favorite <topic>?	3	1
Do you <sentiment>any type of <topic>?	2	0
Do you <sentiment>any <topic>?	2	0
What is your least favorite <topic>?	7	0
Do you have a least favorite <topic>?	8	0

Table A.4: Template questions developed after analyzing the GTKY dataset, with the addition of two questions after a user test

answer	answer_id	same_sentiment	fetch_count	default_negative	default_positive	use_noun	use_sentiment
Yes, I <sentiment><noun>.	1	1	0	0	0	1	1
No, I don't <sentiment><noun>.	1	0	0	0	0	1	1
No, I <sentiment_1><noun>.	1	0	0	0	0	1	0
I <sentiment_1><noun>.	1	0	0	0	0	1	0
I do <sentiment><noun>. I also <sentiment><noun_1>.	1	1	1	0	0	1	1
Yes, I do!	1	1	0	0	0	0	0
No, I don't.	1	0	0	0	0	0	0
I don't <sentiment><noun>but I <sentiment><noun_1>.	1	0	1	0	0	1	1
No, I don't <sentiment><noun>but I <sentiment><noun_1>.	1	0	1	0	0	1	1
I <sentiment><noun_1>, <noun_2>and <noun_3>.	2	1	3	0	0	0	1
I <sentiment><noun_1>the most! But I also <sentiment><noun_2>.	2	1	2	0	0	0	1
I <sentiment><noun_1>.	2	1	1	0	0	0	1
I <sentiment><noun_1>and <noun_2>.	2	1	2	0	0	0	1
I don't <sentiment>any <noun>.	2	0	0	0	0	1	1
I don't <sentiment><noun>.	2	0	0	0	0	1	1
My favorite <noun>is <noun_1>.	3	1	1	0	1	1	0
I like <noun_1>the most.	3	1	1	0	1	0	0
I think I like <noun_1>the most.	3	1	1	0	1	0	0
My favorite <noun>is probably <noun_1>or <noun_2>.	3	1	2	0	1	1	0
I'm not sure but I really like <noun_1>a lot.	3	1	1	0	1	0	0
I don't have a favorite <noun>.	3	0	0	1	0	1	0
I <sentiment_1><noun>, so I can't think of a favorite.	3	0	0	1	0	1	0
My favorite <noun>genre is <noun_1>. But I also like <noun_2>and <noun_3>.	4	1	3	0	1	1	0
My absolute favorite <noun>genre is <noun_1>.	4	1	1	0	1	1	0
I think I like <noun_1>the most.	4	1	1	0	1	0	0
I <sentiment_1><noun>.	5	1	0	0	0	1	0
Hm...I don't really like <noun>that much.	5	0	0	1	0	1	0
I like it!	5	1	0	0	1	0	0
My favorite <noun>are <noun_1>and <noun_2>.	6	1	2	0	1	1	0
It has to be <noun_1>and <noun_2>.	6	1	2	0	0	0	0
I <sentiment_1><noun>.	6	0	0	1	0	1	0
I don't have a favorite <noun>because I <sentiment_1><noun>.	6	0	0	1	0	1	0
My least favorite <noun>is <noun_1>.	7	1	1	1	0	1	0
I do, my least favorite <noun>is <noun_1>.	8	1	1	1	0	1	0

Table A.5: Template answers that are used together with a sentiment memory, to answer what things the agent likes or dislikes.

A.6 Front-End - Question Answering Agent

MemoryDialogueBot

Consent form and information

Description

This research is part of a Master Thesis Project in the topic of memory-based conversational agents. The agent is developed to be a question answering component, answering self-disclosure type questions. This is part of a research study on how people interact with a 'personal question answering' chatbot. After the conversation you will be asked to answer a number of questions. The questions are about rating how you perceive the agent in terms of its natural-ness (human-like), engagingness, etc., on a 5-point scale. You will also be asked to fill in your age and your estimated English proficiency.

Your participation will take approximately 4-7 minutes.

Risks and benefits

All messages sent to the chatbot will be saved and may be used by the current or a future version of the system. The stored messages may be used to extend the existing question/answer retrieval database. Data given in the survey may be presented in a Master Thesis report & presentation. There is no discomfort involved in this study, nor any risks. We cannot and do not guarantee or promise that you will receive any benefits from this study.

Requirements to participate

You must be 18 years or older to participate.
Basic conversational proficiency in the English language is desired.

Participant's rights

Please understand your participation is voluntary and you have the right to withdraw your consent or discontinue participation at any time without giving any reasons. You have the right to refuse to answer particular questions. You have the right to refuse afterwards (within 24 hours) to allow your data to be used for the research. If you wish to discontinue your participation and have your data removed, please remember your User ID: 65.

All your data collected in this test:

- Is anonymous.
- will be stored in such a way that unauthorized access will be minimized.
- will not be disclosed to third parties.

Contact information

Further information or Withdrawal: If you have any questions or wish to discontinue your participation and for your data to be removed, please contact me by e-mail m.lundellvinkler@student.utwente.nl. Remember to include your User ID 65 if you wish for your data to be removed.

Independent Contact: If you are not satisfied with how this study is being conducted, or if you have any concerns, complaints, or general questions about the research or your rights as a participant, please contact the Ethics Committee, Faculty of EEMCS, University of Twente, tel: 053 -489 6719, email ethics-comm-ewi@utwente.nl

Cookies

Cookies are only used to store necessary data for the chatbot system and user test to be able to run smoothly, such as your User ID and message history with the chatbot.

Instructions

Please ask the agent at least 5 questions to try to get to know the agent, as if it was a real person. Please send complete sentences one at a time.

When to stop asking questions to the agent is decided by you. Afterwards, please fill in a short 10 question survey to finish the test.

■ [I have read and understood my rights, I agree to the terms and hereby confirm my consent.](#)

[Continue](#)

Figure A.1: Front page with instructions and terms for the question answering user test.

MemoryDialogueBot

Hello. Feel free to ask anything about me, whether you want to know what food I like or what my hobbies are.

What is your name?

My name is Keith.

What is your name?

I told you before. My name is Keith.

Ask me a question!

■ [I'm done interacting with the chatbot.](#)

[Continue to survey](#)

Figure A.2: Chatbot interaction page, illustrating retrieval based memory.

MemoryDialogueBot

Age

Estimated English proficiency

Have you previously interacted with a chatbot? ☐ Yes ☐ No

	Not at all				Very much so
The chatbot is engaging/interesting to interact with.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The time it takes to get an answer negatively affects my chatting experience.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
It feels as if i'm chatting/talking with a real person.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The chatbot gave contradicting answers.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The chatbot's answers are related/relevant to my questions.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
It feels as if the chatbot's answers are repetitive.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The time it takes to get an answer from the chatbot is acceptable.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
I would like to interact with the chatbot again in the future.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
It feels as if the chatbot has its own personal background and personality.	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Figure A.3: Survey page for the question answering user test.

A.7 Question Answering User Test Interactions

Ideal example

Q: Who are you?

A: I am Keith Francis.

Q: Do you like to watch movies?

A: I love watching movies but I don't have a lot of free time as of late.

Q: Why?

A: I also get to travel a lot, so I get to do a lot of my college

Q: What movies do you like?

A: I like Avengers.

Q: Do you have any friends?

A: I have lots of friends.

Q: What is the name of your best friend?

A: hi friend is gonna be nice vanilla bean pancake... so i almost always keep track of which one i keep... i love vanilla... so I know which i like best.

Q: What are your hobbies?

A: Well I like to cook, if you can call that a hobby and I like to read alot.

Q: What do you usually cook?

A: I cook a lot of different things, like pancakes, sausages, fajitas, stews, etc.

Q: What music do you like?

A: I like rap.

Q: Who is your favorite rapper?

A: Nobody, but I HE has his moments.

Q: Do you have any pets?

A: Yes, a baby geese and a guinea pig.

Q: What are their names?

A: I don't really remember.

Borderline example

Q: Is it going to rain tomorrow?

A: Probably not exactly this morning.

Q: Are you a human=

A: I'm a robot.

Q: why?

A: because its summer so i have to get my legs together pretty quick.

Q: Why are your legs apart?

A: For different things.

Q: What kinds of things?

A: I like to play sports.

Q: Watersports?

A: Yeah, I've seen storms, hail and snow.

Out of scope example

Q: What is 3/0

A: Fridays are for fun, not much major other than I play the violin, sing, and am in college so has lots of family.

Q: Yes

A: Yes, I did.

Q: No

A: No, I'm actually quite happy with Cape Cod.

Q: Yes

A: Yes, I did.

A.8 Questions for Evaluation

Questions	
what do you do for fun?	Do you have any plans for the weekend?
What do you do for a living?	do you have a favorite kind of music you like to listen to?
What do you like to do in your spare time?	Have you always lived in Texas?
what else do you like doing?	How do you like playing sports?
What are you studying?	do you like soccer?
what do you do for fun in Missouri?	Have you seen ANY of the Hunger games movies?
Do you love American food?	Are you old enough to drink?
Are you ready for the snow to melt and spring to start?	Do you ski?
Are you a student at Harvard?	How are you today?
What kind of education do you have?	do you meditate?
Where do you live?	What's your favorite beer?
What type of job do you do?	Are you on Facebook?
Have you seen any good movies yet?	Are you also in college?
Do you cook?	have you ever been to italy?
What do you do with your summer?	Do you speak any other languages?
What kind of music do you like?	What's your favorite kind of food?
What's your favorite cake you have made?	What is your favorite place in the entire country?
do you have any recommendations for things to do in the area?	what is your name?
do you like The Count from Sesame Street?	What do you do outside of school?
how often do you do these types of studies?	What kind of ice cream flavor would you suggest for me?
do you have any pets?	what sport do you play?
Are you into yoga?	are you religious?
What kind of field do you study in?	you like anything to do with computers?
What do you do?	What's a typical day like for you?
Do you like reading?	Have you enjoyed the sunshine?
So are you a student or working?	is there anything fun planned for you this weekend?
Any favourite sports?	what do you do that is important to you in your life?
Have you ever seen snow?	Can I ask your age?
How is the weather where you are?	Do you enjoy reading?
Do you like to watch any sports?	What are you planning to do on your vacation?
do you like to travel?	Have you ever been in Asia?
do you have any hobbies?	do you enjoy swimming?
do you have plans for the weekend?	What do you do at home on a Saturday?
are you from here?	What are your favorite genres then?
Did you study abroad at all or plan to?	What do you do when stressed?
What's your major?	Are you an artist?
What's your opinion on him?	What do you do when you're on the computer?
Do you go to school in Boston?	What's your favorite thing to cook?
You really like to read, don't you?	where in the world do you live?
When did you start singing?	Are you a monk too?
Do you have children?	What do you do with your winter vacation?
How old are your kids?	Did you ever leave the country before?
How's your morning going?	
Do you want to stay in the area or move to another state?	

Table A.6: Questions generated by a model accidentally trained on question asking, that were then used for evaluating answer relevancy of fine-tuned generative models.

A.9 Final User Test Interactions

Sample Interaction 1 - Session 1

User: Hi

Chatbot: SYSTEM: PLEASE enter your email address to continue.

User: REDACTED_EMAIL

Chatbot: Hi! What's your name?

User: REDACTED_NAME

Chatbot: Cool!

Chatbot: What's one sport that you like?

User: I like to swim

Chatbot: My favorite sport is soccer.

User: What do you like about soccer?

Chatbot: I like watching it, it's hard to watch with a computer, but I enjoy it.

Chatbot: What do you like about swim?

User: I like swimming because it's nice to be in the water

Chatbot: Okay.

Chatbot: Here's another question, what's your one favorite pet?

User: My cat

Chatbot: Okay. I dislike cat, but I like sheep

User: do you have a sheep as a pet?

Chatbot: No, Sheep is not my pet. I have a dog.

Chatbot: Thank you REDACTED_NAME! You have now finished the first session, please go back to the form for a small survey, and start a second session in a few days. I will send you a reminder via email.

Session 2

User: Hi

Chatbot: SYSTEM: PLEASE enter your email address to continue.

User: REDACTED_EMAIL

Chatbot: Good day REDACTED_NAME! How are you?

User: I'm great! How are you?

Chatbot: It has been long.

Chatbot: Last time we talked about swim, did you do it recently (<30 days)?

User: No I didn't

Chatbot: You think you will do it in the near future?

User: I hope so!

Chatbot: I see.

User: Do you have any hobbies?

Chatbot: I like to ski, I like playing video games, reading, and traveling.

Chatbot: We also talked about cat. Do you happen to own one?

User: That's nice!

Chatbot: And what color?

User: Yes I do have a cat

Chatbot: Okay!

User: My cat has different colors

Chatbot: lol My cat has a light gray fur.

Chatbot: Great. It was nice talking to you!

Chatbot: Thank you! Please proceed to the form for a small survey to complete the test.

Sample Interaction 2 - Session 1

User: Hi

Chatbot: SYSTEM: PLEASE enter your email address to continue.

User: REDACTED_EMAIL

Chatbot: Hi! What's your name?

User: REDACTED_NAME

Chatbot: Cool!

Chatbot: What's one sport that you like?

User: Yoga

Chatbot: I think my favorite sport is soccer.

User: Do you play?

Chatbot: No, but I do enjoy playing other sports like basketball, hockey and sculls.

Chatbot: What do you like about Yoga?

User: I can do it whenever I want

Chatbot: Okay.

Chatbot: Here's another question, what's your one favorite pet?

User: Cat

Chatbot: Alright. I dislike cat, but I like sheep.

User: Why do you like sheep?

Chatbot: Sheep is soft so I don't mind it, but I enjoy swimming with it.

Chatbot: Thank you REDACTED_NAME! You have now finished the first session, please go back to the form for a small survey, and start a second session in a few days. I will send you a reminder via email.

Session 2

User: Hi

Chatbot: SYSTEM: PLEASE enter your email address to continue.

User: REDACTED_EMAIL

Chatbot: Good day REDACTED_NAME! How are you?

User: I'm fine. How're you?

Chatbot: I'm fine.

Chatbot: Last time we talked about Yoga, did you do it recently (<30 days)?

User: Not since we last talked, but before (<30 days) - yes.

Chatbot: Mh okay.

User: Did you play soccer?

Chatbot: No I played a little bit at the junior college I grew up in, it was very tough.

Chatbot: We also talked about Cat. Do you happen to own one?

User: I do, yes.

Chatbot: And what color?

User: She's tricolor - white, black, yellow.

Chatbot: Mh.

User: So, what're your plans today?

Chatbot: I'd like to go to a movie today.

Chatbot: Great. It was nice talking to you!

Chatbot: Thank you! Please proceed to the form for a small survey to complete the test.