Domain Coverage of Low-Resource Chatbots

Bachelor thesis in Data Science

by

Kasper Thomas Gartside Knudsen
kakn@itu.dk

Supervisor: Christian Hardmeier

IT-University of Copenhagen
Denmark

May 2022

Contents

1 Introduction

Chatbots are software applications that automate conversation using artificial intelligence and natural language processing, mostly employed to virtually assist humans online. In machine learning, they can generally be split into two categories — retrieval and generative, the difference being whether they output pre-defined or generated responses. Retrieval-based chatbots classify user input as questions to one of set responses it has available, while generative-based chatbots use very large conversational datasets to recognize linguistic patterns and generate unique replies based on what the user prompts.

In industry, it is typically retrieval-based chatbots that are used for e-commerce, as, regardlessly, there are limited options customers have when seeking assistance. Additionally, a corporation might not want a generative chatbot interacting with customers, as it could reply with anything and would require much more research and maintenance than its alternative. In research, both types of chatbots receive much attention, as retrieval models have more practicality in a business setting, while generative models are subject to more cutting-edge algorithms, often in the interest of passing the turing-test [8], which is yet to be accomplished. These models are much harder to train and evaluate, as they have very long training times and their qualitative predictions do not easily indicate mathematical accuracy.

In the research field and deployment of generative chatbots, they are typically restricted to certain small domains, in order to serve commercial functionality and evaluate predictions more consistently. Open-domain chatbots require immense amounts of data in order to cover all domains, and usually only exist to entertain or impress users momentarily, such as systems like Cleverbot[1] or Eviebot[2]. This bachelor project proposes to investigate domain coverage of low-resource generative chatbots, uncovering whether open or closed approaches perform better on average, according to their BLEU [7] results, tested on various domains. Simultaneously, multiple iterations of these models will be run with different parameters, in an attempt to uncover what factors that the quality of text generation favour most, considering training time.

[1] https://www.cleverbot.com/
[2] https://www.eviebot.com/en/

2 Background

The field of chatbots extends beyond machine learning methods, early implementations of which leveraged rule and keyword-based methods. The first chatbot is generally accepted to be ELIZA [9], a 1966 system created by Joseph Weizenbaum, which mimicked a therapist by rephrasing keywords of the input into questions. Despite this relatively simple implementation, many early users were convinced of ELIZA's intelligence, inspiring similar models in the coming decades, with the common goal of passing the turing test [8].

In more recent times, the focus of chatbots have been that of ML implementations. These approaches can be categorized into intent-detection (also known as retrieval) and generative types. Retrieval approaches use classification models to predict intents based on questions prompted by the user. Intents are essentially classes representing topics the chatbot is programmed to answer. Since the number of intents are dependent on the programming of the system, there is no variety in the wording of responses. My interest in this field stems from my second semester at ITU, where I created a retrieval-based chatbot like this in my free time. My conclusion to the experience was that building a generative model instead would be a far more interesting and entertaining task.

Generative chatbots create unique responses given a (usually) large conversational training dataset. They do this using an approach called sequence-to-sequence [1], which sequentially "translates" input into a response using statistical or neural machine translation methods (SMT or NMT). The chatbot used in this paper uses the NMT method, employing two recurrent neural networks (RNNs) that work as encoder-decoder pairs. The encoder represents the input as a vector of fixed length, the decoder then translating this into a target sequence of variable length. These two networks are trained at the same time to maximize the probability of a target given its source.

The NMT approach has become the standard for generative chatbots, most of which use encoder-decoder architecture composed of LSTM (long short-term memory) networks applied on Word2Vec embeddings. The chatbot in this project uses this technology, except vectorizing using one-hot encoding. One paper uses stacked LSTM architectures alongside BERT vectorizations to create a domain-specific chatbot from little data [5], reporting BLEU [7] scores of ~ 0.513. Another paper proposes its own two-step training and mixed encoding-decoding methods to train a generative chatbot from little data [6], reporting BLEU scores of 0.5076. This bachelor project uses a rather standard implementation to contrast response generation of open and closed domain models, trained on little data.

3 Methodology

3.1 Data description

The chosen dataset for training the chatbots was the Cornell movie dialogue corpus [3]. It contains 220,567 conversational exchanges from 517 movies, including metadata describing what character, movie and genre of movie is in question. This dataset was especially appropriate for this problem since conversation data is crucial for training conversational chatbots, and the genre feature allows movies to be split into separate domains. This allows for separate models to be trained and tested on specific domains, authorizing the analysis of what factors impact domain coverage the most.

3.2 Building the chatbot

The python code[3] created for this chatbot contains almost 500 lines and uses 18 methods to build, train and test the chatbot in question. The structure of the chatbot in this code was guided by a towardsdatascience article[4], specifically on the setup of the seq2seq model. This section of the paper will briefly summarize the methods used to obtain results.

The first steps were to read, select and clean the data. After extracting move lines that pertained only to the genre in question, the lines were queried to extract question-answer pairs that the metadata claimed were one-on-one conversations. This was done to ensure the chatbot was not trained on false interactions, for instance when a scene ends and a new one begins. From here, the data was stripped of punctuation and replaced words with contractions using Regex, then constricted to only consider uterrances between 2 and 26 words. The pairs were then shuffled to include a wide range of movies, using a random seed to ensure reproducibility when creating test sets. The unique words included in each pair were added to lists of input and target tokens.

The next step was to vectorize this data to be read by the LSTM [4] encoder-decoder model. In order to create one-hot vectors, input and target feature dictionaries were initialized, containing words as keys and indexes as values. To decode sentences, the same were initialized in reverse. Three vectors were made using these dictionaries: encoder input data, decoder input data and decoder output data. These are three-dimensional matrices consisting of vectorizations of a word, listed in a sentence of words, in a list of sentences (utterances). The length of the inner matrix is dependent on how many tokens there are for the given encoder/decoder input. The matrices are filled in accordance to the current line, timestep and token in question, placing a single 1 in the matrix of 0's, distinguishing it from the rest (thus the name one-hot). The encoder-decoder model employs a method called teacher forcing, where the target word is passed as the next input to the decoder.

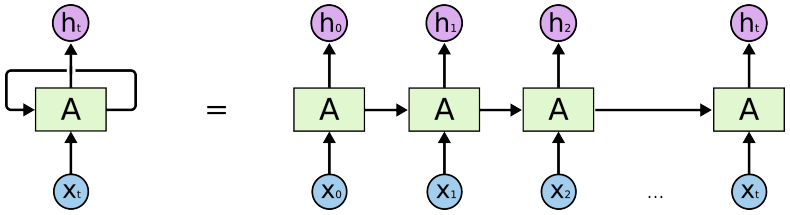

Figure 3.2.1: Teacher forcing[5]

[3] https://github.itu.dk/kakn/thesis
[4] https://towardsdatascience.com/generative-chatbots-using-the-seq2seq-model-d411c8738ab5

This method is not actually used in this step, but provides some context as to why the matrices are being filled as they are. In an RNN, you have multiple repetitions of the same cell. In figure 3.2.1, x_0 is the current input, which gets passed through the cell (the current timestep), producing output h_0, which in turn is used with x_1 to determine h_1, and so on. This is done to remember previous inputs in constructing the final generation, h_t. The generations wouldn't make any sense if they only made predictions on each word independently, so it makes sense why we would want to include the timestep in the vectorization we are passing to our model.

In order for the sequence-to-sequence model to be trained, it first had to be tailored to contain the one-hot matrices. The model itself required encoder-decoder models, each consisting of an input layer for holding the vectors, an LSTM hidden layer and a dense softmax layer (only for the decoder output). These models, including the seq2seq, were set up using the TensorFlow library, Keras [2]. Now that the training model was ready, it was fitted with the one-hot matrices, its weights and metrics being stored locally for later analysis.

In a typical machine learning project, the model would be ready for tests. However, the current model only works if the target sequence is known, and as such needs to be re-written to accommodate applications of a generative chatbot. A new seq2seq model has to be built, as we can no longer prepare for what to decode. The previous encoder can be repurposed as we saved the weights and are able to recreate the model without retraining. However, a new decoder has to be initialized, preserving the architecture from the dense and LSTM layers of the previous model.

From here, all there is left is to create methods that allow our encoder-decoders to read user input and generate a response. The user input is first cleaned with Regex, then vectorized in the same manner as before, and finally passed into the "decode_response" function, that returns a generation. This function first passes the input into the encoder model, receiving output states which are then passed into the decoder. The decoder recursively predicts the most likely output token, updating states and the target sequence until a stop condition is met.

3.3 Investigating performance

Now that the chatbot was functional, it was trained separately twice with one and two thousand random training samples. This was done to investigate how training data affected training time, accuracy and convergence. These two types of models will henceforth be referred to as 'small' and 'large'. While training, the metrics of each epoch were stored locally for analysis of the models' performance. While these metrics do not determine the chatbots conversational ability, they contrast its recognition of seen and unseen data, which is rather useful for evaluating the potential domain coverage of a chatbot.

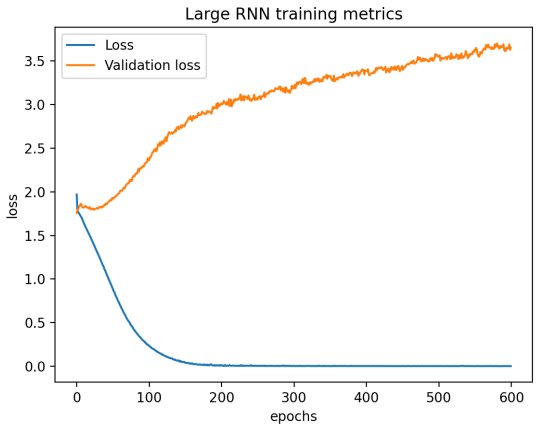

Figure 3.3.1: Training performance of large open-domain chatbot

The training time of this larger model was almost double to that of the smaller one, so it seemed interesting to investigate whether training time favoured more epochs or samples when analysing domain coverage. The hypothesis here was that more samples would result in better domain coverage, while more epochs would result in greater response quality. The smaller models epochs were therefore doubled to 1200. Since this model was trained merely to test the functionality of the chatbot, it did not matter that it was so clearly overfitted. The rising validation loss did indicate that the model didn't need this many epochs, but this was likely due to the diversity in training data, given there was no specific domain. Later visualizations of domain-specific chatbot training found high variance in the relationship between epochs and loss/accuracy convergence, likely due to the varying quality and content of said domains, so this factor was kept static within model sizes to draw fair comparisons of results.

3.4 Defining domains

For the purpose of investigating how data affected domain coverage of the chatbot, six domains were defined, each representing a different genre depicted by the Cornell movie dialogue corpus. The choice of genre was dependent on how many movies were solely classified as said genre, as a movie having multiple genres may implicate bias in the results. Initially, only four genres had sufficient data to train these models; comedy, drama, horror and thriller. Fearing that these were not diverse enough, a fifth domain, 'sci-fi' was manually sourced by what I determined to represent sci-fi accurately (e.g. Star Wars, Star Trek, etc.). The results of this domain were expected to differ from the others as it contained multiple genres, and as such could test a hypothesis that such a model may be biased. In order to test against the generalization of these five domains, a 'general' domain was initialized that contained data from all five genres. The hypothesis of this domain was that it would perform worse on individual genres, but have a greater average performance.

domains	general	comedy	drama	small horror	horror	thriller	sci-fi
vocab-size	**12243**	4398	7379	*2058*	3748	3153	3273
corpus-size	**23803**	3890	11584	*1383*	3539	2697	2231
type-token	*0.5143*	1.1306	0.6386	**1.4881**	1.0591	1.1691	1.4671
avg-utterance	9.5367	9.7721	9.5199	9.2748	*9.2204*	9.4703	**9.7745**

Figure 3.4.1: Summary statistics of domains

Figure 3.4.1 statistically analyses the different domains, which will be useful in evaluating later results. 'Vocab-size' counts the number of unique tokens per domain, 'corpus-size' counts the number of question-answer pairs, 'type-token' is the ratio between these two, and avg-utterance calculates the average length of questions/answers or *utterances*. Bold and italic entries represent the highest and smallest results per metric, respectively. Each of these domains were used to train one 'large' and one 'small' model, each. The sole exception is the horror chatbot, which did not originally have enough samples to satisfy this 'large' model. The solution was to inflate the large training set by adding an exception that a movie having 2 genres, one of which was horror, would be added to the training set. This difference was expected to be noticeable in the results. For the twelve models, other parameters such as grid-size, LSTM hidden layers, optimizers etc. were kept static as they were not feasible to grid search, considering the long training time. The uniform hyperparameters across models would then also highlight the importance of training data quality when evaluating text generation and domain coverage. The results of these large and small versions were expected to contrast the trade-off between training size and iterations.

3.5 Evaluation

In order to evaluate the domain coverage of each model, the BLEU [7] metric was used to evaluate the similarity between a sample answer and the bots' response to the sample question, computing a ratio of words seen in both texts. Standard BLEU implementations account for n-gram similarities, typically 4-grams. However, the results of using this approach resulted in scores of 0, so a smoothing method was applied to the 'sentence_bleu' function, in order to bypass harsh behaviour when no n-gram overlaps were found. Because of the selected evaluation metric, it was expected that the best performing models would potentially have the largest vocabulary size. The results were also suspected to be indicative of the average sentence lengths per domain, as having more words per generation would reduce the chance of getting a score of 0. Other metrics such as *perplexity* and manual evaluation were considered for evaluation, but scrapped as they (respectively) didn't address domain coverage and were too time-consuming.

Once the twelve models had been trained, their bleu-scores were calculated using the remainder of samples not included in each training set. The results would then be two matrices depicting the average bleu-score of each model on each test set, with additional averages on either axes to determine which genre generalized the best. Unfortunately, the size of the small horror training set meant there were only 231 unused conversations left over. In order to represent a fair evaluation across the board, all tests were conducted with 231 test samples.

To justify whether the BLEU implementation accurately determines domain coverage, its results will be compared with several metrics performed on the domain training sets, as well as the vocabulary overlap of

said domains. These results may help in contextualizing the BLEU results, in the case they are not indicative of text generation quality or domain coverage. Figures 3.5.2 and 3.5.4 present the vocabulary overlap in percentages, the averages of which do not consider the identity line. For instance, '47.82%', shown in row 2 column 3 of figure 3.5.2, depicts the percentage of the comedy domain vocabulary that is found in the general domain vocabulary. Bold and italic entries represent the highest and lowest percentages for the domain in question. Alternatively, the validation losses of the best and worst models according to the BLEU results will be plotted to justify whether they indicate model quality. This comparison can only be drawn on models with their own domains, as this is close to what their validation loss indicate.

In the case that the bleu-scores do not correlate with domain coverage nor response quality, some qualitative results will be presented as a fail-safe. The models the BLEU scores deem best and worst per size type will be prompted a few basic questions, to demonstrate their basic conversational abilities.

small	general	comedy	drama	horror	thriller	sci-fi
vocab-size	**2469**	2426	2294	*1791*	2013	2236
corpus-size	1000	1000	1000	1000	1000	1000
type-token	**2.469**	2.426	2.294	*1.791*	2.013	2.236
avg-utterance	9.537	9.838	9.529	*9.309*	9.484	**9.864**

Figure 3.5.1: Summary statistics of small domains

small	general	comedy	drama	horror	thriller	sci-fi	average
general	100%	47.82%	52.7%	**55.05%**	51.71%	*46.69%*	**50.79%**
comedy	46.98%	100%	44.59%	**50.64%**	45.6%	*39.94%*	45.55%
drama	48.97%	42.17%	100%	**49.97%**	44.61%	*40.21%*	45.19%
horror	39.94%	*37.39%*	39.01%	100%	**41.53%**	37.79%	*39.13%*
thriller	42.16%	37.84%	39.15%	**46.68%**	100%	*37.48%*	40.66%
sci-fi	42.28%	*36.81%*	39.19%	**47.18%**	41.63%	100%	41.42%
average	44.07%	*40.41%*	42.93%	**49.9%**	45.02%	40.42%	43.79%

Figure 3.5.2: Vocabulary overlap of small domains

large	general	comedy	drama	horror	thriller	sci-fi
vocab-size	**3812**	3433	3368	2969	*2812*	3133
corpus-size	2000	2000	2000	2000	2000	2000
type-token	**1.906**	1.7165	1.684	1.4845	1.406	1.5665
avg-utterance	9.7375	9.793	9.5212	*9.185*	9.5508	**9.8032**

Figure 3.5.3: Summary statistics of large domains

large	general	comedy	drama	horror	thriller	sci-fi	average
general	100%	54.27%	58.28%	55.98%	**59.5%**	*53.11%*	**56.23%**
comedy	**48.87%**	100%	44.33%	46.48%	48.36%	*42.52%*	46.11%
drama	**51.5%**	43.49%	100%	47.22%	48.44%	*42.8%*	46.69%
horror	43.6%	*40.2%*	41.63%	100%	**46.19%**	41.56%	42.64%
thriller	**43.89%**	*39.62%*	40.44%	43.75%	100%	40.76%	*41.69%*
sci-fi	43.65%	*38.8%*	39.82%	43.85%	**45.41%**	100%	42.31%
average	46.3%	*43.28%*	44.9%	47.46%	**49.58%**	44.15%	45.94%

Figure 3.5.4: Vocabulary overlap of large domains

4 Results

4.1 Quantitative results

In the following tables, rows represent models and columns represent test data. For instance, in figure 4.1.1, the second row and third column shows the score '0.497', which is the BLEU score of the general model tested on comedy data. A **bold** score represents the highest score for a model, and an *italic* score represents the lowest. The same rules apply for the averages on either axes, the top right indicating averages per model, and bottom left indicating averages per domain test set.

small	general	comedy	drama	horror	thriller	sci-fi	average
general	0.0479	0.0497	0.0513	0.0566	**0.0572**	*0.041*	0.0506
comedy	0.0545	0.0536	0.0463	0.0542	**0.0601**	*0.0451*	0.0523
drama	0.046	0.0457	0.0432	0.0488	**0.0539**	*0.042*	0.0466
horror	0.0412	0.0431	*0.0367*	0.0445	**0.0567**	0.0468	*0.0448*
thriller	0.0547	0.0551	0.0496	0.0494	**0.0612**	*0.0476*	0.0529
sci-fi	0.0529	0.0554	*0.0524*	0.0551	**0.0571**	0.0555	**0.0547**
average	0.0496	0.0504	0.0466	0.0514	**0.0577**	*0.0463*	0.0503

Figure 4.1.1: BLEU scores of small models

large	general	comedy	drama	horror	thriller	sci-fi	average
general	**0.0577**	0.0494	0.0464	0.0522	*0.046*	0.0474	0.0498
comedy	0.0481	0.0553	0.0491	**0.0578**	*0.0453*	0.0495	**0.0509**
drama	**0.0563**	0.0512	0.0465	0.0469	0.0407	*0.0398*	0.0469
horror	**0.0508**	0.0478	0.0476	0.0477	0.0417	*0.0403*	0.046
thriller	0.0486	0.0521	0.0452	**0.0532**	*0.0346*	0.0367	*0.0451*
sci-fi	0.0491	0.0478	*0.0366*	0.0498	0.0501	**0.0535**	0.0478
average	**0.0518**	0.0506	0.0452	0.0513	*0.0431*	0.0445	0.0477

Figure 4.1.2: BLEU scores of large models

While these results are disappointingly invariant, there still seems to be evidence of correlation.

4.2 Qualitative results

The results of the small models found the sci-fi chatbot generalized best and the horror bot worst. The large model results found the comedy chatbot generalized best and the thriller bot worst. Each of these chatbots were prompted the same basic questions, under the hypothesis that the better generalizing models would converse more realistically.

Small sci-fi chatbot	Small horror chatbot	Large comedy chatbot	Large thriller chatbot
> Human: Hello > Bot: Good! Do you know you know all > Human: How are you > Bot: Too bad.. bad accident.. > Human: Who are you > Bot: It is choi > Human: I am sorry > Bot: This is a action > Human: Good bye > Bot: Hi, sir..	> Human: Hello > Bot: Where? > Human: How are you > Bot: Come on, claudette. Gulp. > Human: Who are you > Bot: I guess > Human: I am sorry > Bot: Ned? We are gonna be working together > Human: Good bye > Bot: Thank you?	> Human: Hello > Bot: Jesus morris > Human: How are you > Bot: Go. > Human: Who are you > Bot: Go? > Human: I am sorry > Bot: I can like the graf on the. > Human: Good bye > Bot: Swear.. I are not?	> Human: Hello > Bot: It paddling > Human: How are you > Bot: Transplants? > Human: Who are you > Bot: He might is not not > Human: I am sorry > Bot: Stop saying you are a information > Human: Good bye > Bot: Transplants?

Figure 4.2.1: Conversations with best/worst performing chatbots

Clearly, all bots lack conversational ability, none of which would pass the turing test. Surprisingly, the smaller chatbots seem to make more sense.

5 Discussion

Evidently, the BLEU scores are very low in both matrices, the highest of which being about a 6% average similarity between the generation and its target. Comparing these results with the $\sim 50\%$ scores reported in similar papers [6][5], clearly something went wrong with the setup of the chatbot or BLEU scoring. Relatively low scores were expected in this paper, considering the chatbots can't feasibly predict a movie quote they haven't seen. However, it is the low variance of these results that are disappointing, falling into 3-7% similarity. I had hoped there would be a more obvious correlation between the models and the genres they were tested on, specifically that they would score highest on their own domain. While this is not the case in most instances, there is still evidence that the results are not essentially random.

The results of the smaller bots show a clear correlation — that the thriller data seemed to be the most recognizable. This unfortunately does not say much about the chatbots, but rather the thriller genre itself. It seems rather likely the thriller genre sample has fewer OOV (out-of-vocabulary) tokens than other test sets, leading to the models predicting the correct word more consistently. This is somewhat confirmed in figures 3.5.1 and 3.5.2, reporting the second-lowest vocabulary size and the second-highest average vocabulary-overlap.

Interestingly, the thriller data did exceptionally poorly in the larger set of models, despite having the smallest vocabulary size and highest vocabulary overlap. Considering this data was used to train its chatbot, it makes sense that the thriller bot reported the lowest average overlap and BLEU score. These indications seemingly accept the hypothesis that more diverse training data leads to better results, further confirmed by the horror and general domains reporting the highest BLEU averages.

The comparison of the two BLEU tables indicate that varied domains test better on large models, while basic domains test better on smaller models. This makes sense as the larger models contain more data and thus have larger domains. The average bleu-score (bottom right) of both tables indicate the trade-off between epochs and training data favours more epochs by a relatively significant 2.3%. This is potentially due to smaller datasets giving the chatbots lesser options to compare the prompts with, the increased epochs enforcing their distinction. This is confirmed further by the qualitative results, showing the large models responding in gibberish.

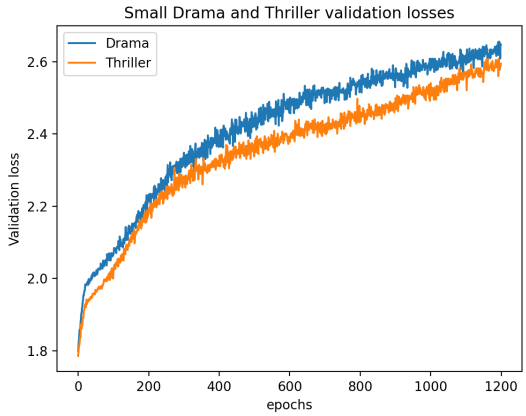

Figure 5.1.1: Smaller models

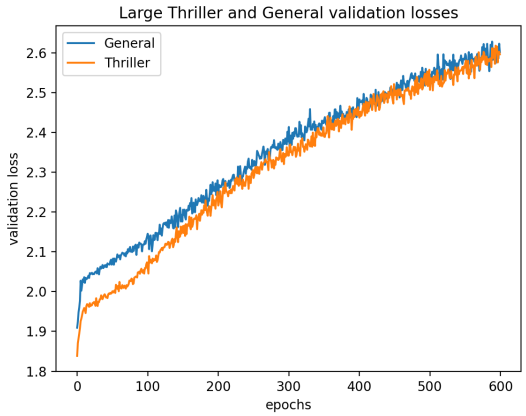

Figure 5.1.2: Larger models

It is generally unclear whether the low scores are more indicative of chatbot quality or applicability of

the evaluation metric, therefore, tests were conducted to validate the metrics consistency. Figures 5.1.1 and 5.1.2 visualize the validation loss of the models that tested best and worst on their own domains. The 'small' results indicate that the bleu-scores do correlate with the model predictions, as the thriller bot clearly has a lower validation loss throughout training. This indicates that bleu-scores may accurately represent domain coverage, but unfortunately this indication is not represented in the 'large' results. The general model clearly predicts unseen samples less consistently than thriller, which is certainly not what the results show. However, validation loss could be expected to be worse with the combination model, as the unseen sample could be from 5 separate domains, instead of just one. This exception is a bit of a stretch, so these two figures don't conclude much about the suitability of bleu for this evaluation.

Another way of evaluating the authenticity of the BLEU scores is to re-train and re-test a model on the same parameters, evaluating whether their correlations remain the same. Since the seq-seq chatbot will have new weights, they will make slightly different predictions. If the variance is high between these predictions, this may indicate that the BLEU metric doesn't represent domain coverage nor generation quality properly. The large general-domain chatbot was retrained twice for this experiment, shown in figure 5.1.3.

	general	comedy	drama	horror	thriller	sci-fi	average
original	**0.0577**	0.0494	0.0464	0.0522	*0.0459*	0.0474	0.0498
retrain-1	0.0533	**0.0558**	0.0524	0.0539	*0.0493*	0.0517	0.0527
retrain-2	0.0484	**0.0566**	0.0511	0.0463	*0.0432*	0.0458	0.0486
std	0.0038	0.0032	0.0026	0.0033	0.0025	0.0025	0.003

Figure 5.1.3: Training performance of large general-domain chatbot

The reruns depict low standard deviation across predictions, even retaining the correlation of low test scores with the thriller dataset. However, the results no longer depict the general domain as the highest scoring test sample, instead choosing comedy as the best fit. While this inconsistency indicates that other results may not be properly represented, this interpretation also makes sense as the comedy domain shares almost 50% of its training vocabulary with the general domain. It could also be argued that the general model would test better on the comedy domain, as it has a much smaller vocabulary size according to figure 3.4.1, however, this argument could be used for any given test domain other than general. This table therefore tarnishes the authenticity of the BLEU results, but does not invalidate its averages or correlations in accordance with other tables and figures.

6 Conclusion

The results concluded the chatbots favoured epochs over training data in terms of text generation quality. It seems that less iterated, more data-heavy models failed to properly converse, despite having more information on the domain. At the same time, the evaluation metric used to compare these had many limitations, specifically in their evaluation of conversational ability, but the results confirm my own experience of conversing with the chatbots — that less data improves conversational ability. The smaller the dataset, the more concise and realistic the responses were, likely since there were fewer options. The larger models seemed to have almost too much data to consider, resulting in short and often nonsensical generations. However, the smaller models responded with a default, confused response more often, indicating they did not understand the input. Ideally, a conversational chatbot should be trained on fewer, high quality conversations than many of varying quality.

In terms of domain coverage, the quantitive results favoured large models, at the expense of the quality of text generation. The large models saw higher scores on the diverse test set, likely due to their increased vocabularies, yet saw less consistent prediction averages. Open-domain chatbots did not necessarily generalize better than closed approaches, their scores laying in-between the best and worst models. However, relative to the tarnished accuracies in larger models, they saw considerable improvements with more training samples. The larger model was able to get the second-highest model evaluation, seemingly indicating that open-domains require large amounts of data to generalize well, as otherwise they are bottlenecked by their large vocabulary sizes and sample variety.

7 Future work

The expensive training and large datasets led to a lot of wasted time in this project, as the models often had to be scrapped for various reasons and retrained. This is time that could have been used to investigate lowering the validation loss of the models by testing different combinations of hyperparameters and alternate RNN's for this problem. Additionally, saving predictions locally during evaluation could have opened the possibility of experimenting with different evaluation metrics, since these tests would not have had to been re-run. Better time management could have led to more accurate models and therefore more concise results and conclusions. There were many oversights that could have been avoided with better planning, but this was especially difficult being the only contributor to the code of this large-scale project. In the future, I would be sure only to use domains that cover common conversational topics, merging them and grid-searching for optimal hyperparameters to minimize the validation loss.

The results of this project inspired me to come up with an alternate approach to developing conversational chatbots — a combination of a linguistic and generative approach. It would use the technology of linguistic (retrieval) models, e.g. finding the most likely answer to a question posed by the user, but this answer would then be sourced from a large dialogue corpus, instead of select replies determined by the developer. This would combine the accuracy of the retrieval model (as replies are not generated) and the domain coverage of a generative chatbot (large dataset). So whatever the user inputs would get a coherent response.

8 Appendix

References

[1] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[2] Francois Chollet et al. Keras, 2015.

[3] 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.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Jurgita Kapočiūtė-Dzikienė. A domain-specific generative chatbot trained from little data. *Applied Sciences*, 10(7), 2020.

[6] Jintae Kim, Hyeon-Gu Lee, Harksoo Kim, Yeonsoo Lee, and Young-Gil Kim. Two-step training and mixed encoding-decoding for implementing a generative chatbot with a small dialogue corpus. In *Proceedings of the Workshop on Intelligent Interactive Systems and Language Generation (2IS&NLG)*, pages 31–35, Tilburg, the Netherlands, November 2018. Association for Computational Linguistics.

[7] 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*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[8] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[9] Joseph Weizenbaum. Eliza a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, January 1966.