Mimic: Character Based Chatbot

Nirvan S P Theethira
nith5605@colorado.edu

Charlie Carlson
chca0914@colorado.edu

Ketan Ramesh
kerk1228@colorado.edu

Pramod Venkatesh Kulkarni
prku8035@colorado.edu

Abstract

In this project we explored the problem of creating a chatbot that could mimic a popular television character's personality, Joey from Friends. Our dataset was extracted directly from the Friends scripts from each season. We worked with two different neural network models which we trained on the same dataset: a sequence-to-sequence model and a transformer model. To evaluate our models' performance we considered automatic metrics often used to evaluate translation bots and a human based metric inspired by the Turing test. Not surprisingly, neither model performed well with respect to the automatic metrics. Moreover, we found that both models failed to consistently convince a human tester that their responses captured the personality of Joey. We review a collection of troublesome instances for each model and theorize ways in which each model could be improved. We conclude that both models had only minor success but it may be interesting to experiment with slightly more complex models and different datasets.

1 Introduction

A chatbot is a program that provides conversational output in response to user input. They have many applications such as customer support interfaces, general question answering services, translation apps and virtual assistants. A common goal for chatbots is to simulate a human like interaction for the user. To this end many researches have investigated creating chatbots with "personality" or "identity" [1, 2, 3, 4, 5, 6]. Such chatbots could drastically improve improve the user experience of nearly all chatbot applications. Consider for example the marketability of a witty virtual assistant or an comforting customer service support chatbot.

Similar to [5, 1, 6], we investigate the related problem of training a neural network to respond like a popular television character, Joey from Friends. While general personality qualities might have more wide spread usefulness in real life applications, the ability to mimic well-known personalities such as Joey would also have commercial applications. More importantly, such a task can be seen as an important first step to solving the more general problem of training a machine to behave like a human. Motivated by this and previous examples, in this work we investigate training two different neural network models act as chatbots with personality. In the following sections we will review the dataset we used, the evaluations we performed, the models we considered and then conclude with our results and final thoughts.

2 Dataset and Evaluation

In this section we review what datasets we used, how we created them and how we evaluated the performance of our models on them.

Preprint. Under review.

Figure 1: Joey from Friends televisions show. Played by Matt LeBlanc. Image download from Wikipedia and originally from Friends' press release.

```
The One Where Monica Gets a New Roommate (The Pilot-The Uncut Version)
Written by: Marta Kauffman & David Crane
Transcribed by: guineapig
Additional transcribing by: Eric Aasen
(Note: The previously unseen parts of this episode are shown in blue text.)

[Scene: Central Perk, Chandler, Joey, Phoebe, and Monica are there.]

Monica: There's nothing to tell! He's just some guy I work with!

Joey: C'mon, you're going out with the guy! There's gotta be something wrong with him!

Chandler: All right Joey, be nice.  So does he have a hump? A hump and a hairpiece?

Phoebe: Wait, does he eat chalk?

(They all stare, bemused.)
```

Figure 2: Example script content from Friends dataset.

3 Dataset

To create the personality based chatbot, we first needed a personality to mimic. Ideally the personality would be familiar to a large number of people and there would be a large amount of freely available data. For these reasons we starting looking at characters from popular television shows with multiple seasons. Moreover, we surmised that the show's writers would have developed a strong and identifiable character personality over the span of several seasons. We expceted such a personality to be much easier to mimic. We also considered movie characters but ultimately, the data available for a movie character was sufficiently smaller than that of a television show character.

For this project, Friends was chosen as the television show and Joey Tribbiani was chosen as the character. We choose Friends because the data was available, the show is popular and it has many seasons. Joey was chosen to mimic specifically because as has a very distinctive personality in the show. For those not familiar the show, Joey can be described as naive, sarcastic, loving, misogynistic, promiscuous and loud.

The raw data set was retrieved from: (https://fangj.github.io/friends/). The scrips were split up by episodes with each episode having an associated comma-separated values (CSV) file which held its script. Each script indicated the character speaking and the dialogue they spoke along with scene discriptions and stage directions (See Figure 2). Each script was parsed to extract *statement response pairs* from each scene (See Figure 3). A statement response pair is two dialogues from the script

	Charachter Input	Charachter Output
0	Ooh... I love Barbados!	Ooh... I cant believe Im kissing you. Im kissi...
1	Hey, you know, before you said that nothing co...	Well, I only said that because of Ross, you kn...
2	Rach, you there?	Oh my God, its Ross. What are we gonna do?
3	Oh, ju-ju-just stay calm. Just be calm. For al...	No idea what it means.
4	Yeah, sure...	I dont believe this... Have you guys been...
5	You would think! Joey!	Is he gone?
6	How are you doing this?	Pssst...
7	Oh no... Have you thought about it how complic...	Well, hes with Charlie now.
8	Yeah, but he wants to talk to you before anyth...	Youre a pain in my ass, Geller!

Figure 3: Example of statement response pairs extracted from Friends dataset.

	Generic Questions	Generic Answers
0	What's up?	Not much.
1	you are not put together	i have always thought whoever did it could hav...
2	Tell me a joke	what do you get when you cross a road and a st...
3	Who uses super computers?	Anybody who wants to work with large numbers ...
4	how far is the sun	the sun is about 93 million miles from earth.
5	leo tolstoy	Is one of Russia's greatest writers.
6	what is chemistry	the science of mixing chemicals.
7	Bend over	That's personal!

Figure 4: Example of statement response pairs extracted from the generic questions and answers dataset.

made by two different characters (*stater* and *respondent*)the second (*response*) immediately following the first (*statement*). We then collected all pairs where Joey was the respondent. We called this the Joey dataset.

As a prerequisite to training a model to mimic Joey's personality was training a model to sound human in general. To this end we wanted to pretrain both our models on an additional dataset of general question and answer pairs. Ideally this would help both models to form more proper English sentence and respond appropriately to questions made by the stater. Around 500 generic data points were used for training (See Figure 4). We called this the generic question and answer dataset.

We randomly selected 20 percent of each dataset to be used as testing data and used the remaining data as training data. After training and testing our models we realized that there were a few things we could have done to significantly improve our overall experiment. We discuss in a later section about what impacts we think the lack of these actions had on our final results.

One key issue with your Joey dataset In the data extraction process outlined above, we describe collecting using the previous dialogue to any Joey dialogue as a statement. However, it is easy to see that this is not always ideal. Indeed, as a result of this, every time a person spoke, regardless of weather Joey heard it or not, Joey's dialogue was considered a response. Consider the case that Joey enters a scene halfway through and says "hello everyone", this will be recoreded as a response

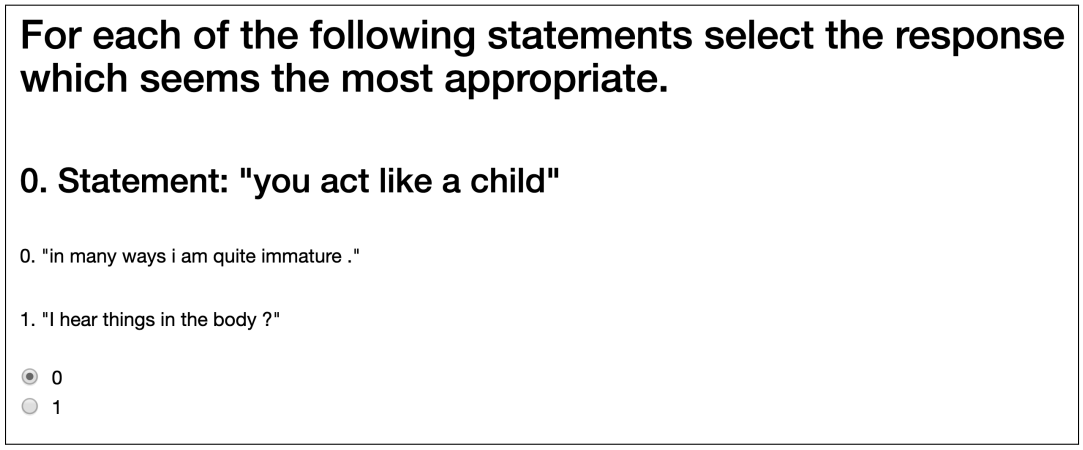

Figure 5: Example of a question for the human evaluation metric.

to anything said before his entrance. Eliminating this problem by implementing a better extraction algorithm could lead to a better dataset and thus better results.

Another concern is that our generic question and answer dataset was too small and not as generic as desired. Indeed, more data is always nice but also finding a corpus that had a lot more examples of normal human communication would be ideal. We believe this process of pretraining on some larger generic dataset would help give each model a better understanding of English.

3.1 Evaluation

To evaluate the performance of our models we considered two types of metrics: a collection of automatic metrics and a human metric.

The automatic metrics we considered were bilingual evaluation understudy (BLEU), recall-oriented understudy for gisting evaluation (ROUGE), metric for evaluation of translation with explicit ordering (METEOR) and word error rate (WER) [7, 8, 9]. These metrics are often used to evaluate the performance of translation bots and in some cases other chatbots. However, despite a strong relationship between chatbots with personality and translation bots, these metrics are believed to be poor measures of performance for personality chatbots [10, 11, 12, 13]. Despite this, they are still often used to evaluate some aspects of a chatbot's performance.

The idea behind each of these metrics is to compared a response generated by one of our models against a saved test response. The general idea behind BLEU, ROUGE and METEOR is to measure the overlap of words in the generated response and the test response. The BLEU score is the normal BLEU score computed with weights $(.25, .25, .25, .25)$. For ROUGE we report the F1 score computed after calculating both the precision and recall. We consider both ROUGE-1 and ROUGE-ℓ. We also collected additional ROUGE scores (e.g. ROUGE-4) but they were approximately 0 for all of our experiments. We report the normal METEOR score. All of these metrics output a score between 0 and 1 with 1 being the best. The WER metric computes the word edit distance between the generated response and the test response and outputs this divided by the length of the test response. For the WER score, a small number was best but the numbers could range to be arbitrarily large (if the generated response was very large compared to the test response). We then average these scores over our entire dataset except for ROUGE which uses a corpus calculation. We refer the reader to the appropriate references for a detailed explanation of these metrics.

For the human based evaluation we designed a jupyter notebook which randomly generated questions for each model from each dataset. Each question had a test statement, test response and generated response. The responses were placed in random order so that the tester would not know which was the generated response. The tester was asked to select which response was most Joey like or appropriate for Joey to say given the response (See Figure 5). We considered 4 testers familiar with Friends and the Joey character. Each tester answered at least 30 of these questions for each model and dataset combination. For each tester and each combination, we calculated a final score by taking the total

fraction of questions where the generated response was selected. We then averaged them to get our final score for the combination. Like the classic Turing test, an ideal outcome would be a score of .50. That is, our model would create responses as Joey like as those from the actual show and the user would ultimately have to randomly pick between.

4 The Sequence-to-Sequence Model

The first model we considered was a seq2seq (sequence-to-sequence) model. It resembles the one mention in [9]. While this architecture is usually used for language translation, it happens to also be widely used for chatbot creation. The success the seq2seq model has had in other chatbot implementations we researched, along with it's easy implementation in Keras is why this architecture was chosen. In this section we discuss how we developed, trained and analyzed this model.

4.1 Data Preparation

Our first step was further reprocessing our datasets so that we could train on it. We decided to do the following preprocessing steps: We broke the dialogue into words and punctuation mark. A word here does not have to reference and actual English word but a maximal sequence of characters excluding punctuation marks and white space. We decided that treating the punctuation marks as individual tokens would hopefully allow our model to better understand questions, statements and commands. We also removed numerical data since smaller numbers were consistently represented by their word form and larger numbers seemed too rarely appear in the raw data. This seemed like a easy way to reduce our vocabulary. Finally we add a start and end tags to each statement.

To make the above data ingestible for the embedding layer of our model and easier to work with in general, every word, punctuation mark and tag was mapped to a numeric token. The *tokenizer* consumes the entire corpus to create this mapping which is then applied to every statement and response in our dataset to create a collection of input and output sequence pairs. The next was *padding* the tokenized sequences with zeroes so they all have the same size. The input sequences were pre padded while the output was post padded.

Next, every token in the sequence had to be given a unique vector representation [14]. This was handle by a **keras.layers.Embedding** layer. To provide contextual representations of tokens (words, punctuation marks and tags), **GloVe 300-dimensional word embeddings** were used to initialized the embedding layer weights. Since the model outputs a *softmaxed* prediction on each token, the next step involved converting output tokens to *one-hot vectors*. Since the goal of the decoder was to predict the next token given the current token, the tokenized output sequence was shifted to the left by one step before being converted to one-hot vectors.

4.2 Model Architecture

The model consists of an Encoder and a Decoder. Padded input token sequences are converted to a sequence of word embeddings using a **Keras.embedding.layer** as described above. Each of these word vectors are fed to the **keras.layers.LSTM** *encoder long short-term memory* (LSTM) one at a time. The encoder LSTM tries to capture the essence of the encoded input sequence in two *thought vectors*, the *output context vector* and the *hidden state vector*. Both thought vectors are then passed to the decoder (See Figure 6).

The *decoder LSTM* is fed a padded sequence of output word embeddings along with the two thought vectors. The decoder model is trained to use a *dense SoftMax* output layer **keras.layers.Dense** with activation **keras.activations.softmax** to predict the most likely output word among all words in the vocabulary. Once the model is trained, an *inference encoder and decoder model* is created using the trained model weights. The inference encoder model takes in statement and two generated state vectors. The input state vectors are fed into the inference decoder model along with a sentence start tag. All words predicted by the SoftMax layer are captured until an end tag is generated (See Figure 7).

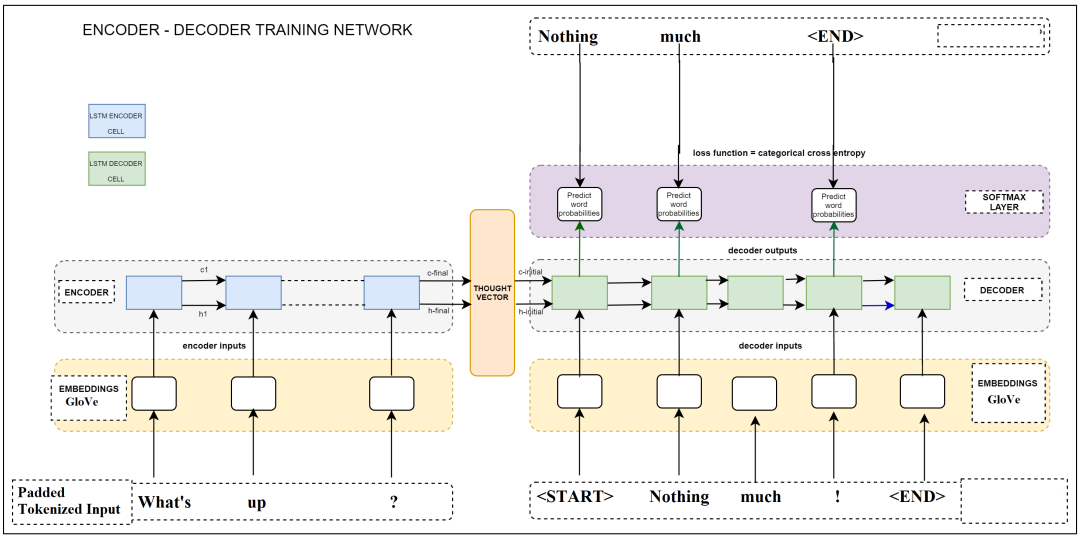

Figure 6: A diagram of our seq2seq model. Image edit from original image found `https://github.com/samurainote/Automatic-Encoder-Decoder_Seq2Seq_Chatbot`.

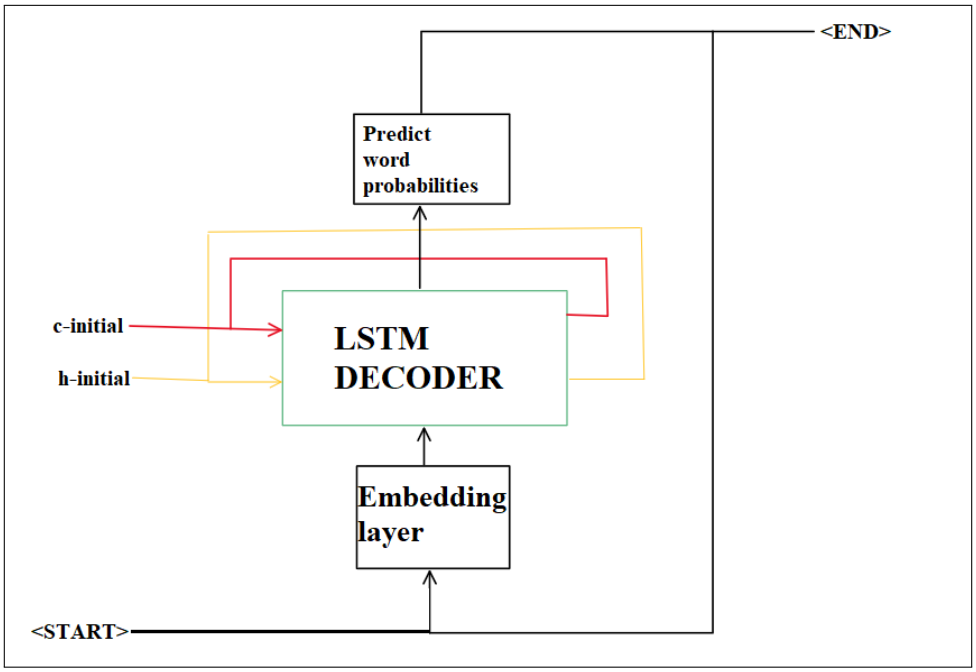

Figure 7: A diagram of our seq2seq model's decoder.

Dataset	BLEU Avg. Score	ROUGE-1 Score	ROUGE-ℓ	METEOR Score	WER Avg.	Human Eval.
Q&A	0.0530	0.213	0.204	0.092	1.554	0.413
Joey	0.015	0.141	0.131	0.064	1.922	0.350

Figure 8: Table showing various scores for this sequence-to-sequence model on the Joey and generic questions and answers dataset (Q&A).

Question	Answer
Do you like food ?	I'm a software , I am an actor . I am , I am what am what am ? ! I am die of error ?

Figure 9: Failure in generating grammatically correct sentence.

4.3 Performance and Error Analysis

As seen in Figure 8 this model did not perform well with respect to most of the automatic metrics. This is somewhat to be expected though given that these types of metrics are known to be flawed when evaluating our kind of task. We will discuss this in greater detail in Section 6. We also note some success in this model for the human evaluation. Again, these numbers may be slightly higher than expected due to the lack of preprocessing of our original dataset. That is, some of it may be based on the human tester randomly picking between two poor responses to a statement and not necessary two equally good responses. We will also review this in greater detail in Section 6.

Here will review a few particular cases where this model consistently failed to perform well on human metrics and follow with some ideas for future improvement.

From Figure 9 it is seen how the model fails to generate grammatically correct sentences. This is because, while the model understands how to mimic Joey, it seems to fail at understanding English grammar. This can be fixed by adding a *conditional random field (CRF)* layer that uses the Viterbi algorithm. More detail is given about this in the next section.

4.4 Pitfalls and Improvements

One major pitfall faced while developing the seq2seq model was during training of our model. Specifically we initially had trouble providing the model with one-hot output vectors. As discussed above, the model uses a output softmax layer to predict the best possible word output. This means the length of each training output vector equals the length of the vocabulary of the entire corpus. This is multiplies by the number of tokens (words, punctuation marks and tags) in the sequence and this is then further multiplied by the number of training data points. With a vocabulary size of 3000, a sequence length of 100 and training data size of 7000, the size of the entire output training data become $3000 * 100 * 7000$. This is too large of a matrix to store in memory. This problem was fixed by training in batches. This was implemented using the Keras $fit\_generator$ function along with our own custom data generation function.

One way we were able to imporove our model's performance was to change how we do our token embedding. The initial model randomly generated embedding vectors of tokens. This did give good results as the word embedding held no information regarding the tokens they represented. This problem was fixed using GloVe word embedding that contextually represented the words they were associated to.

While the current model seems to learn Joey like responses to some extent, it fails to understand how English sentences are structured. This leads to a lot of generated responses failing tests as they are eliminated for not looking like English sentences. We believe this can be fixed by adding a *conditional random field (CRF)* layer using the Viterbi algorithm. Hopefully this would give the model a better sense of English sentence structure.

Another issue we noticed was that this model currently takes an argmax of the output predictions to choose the most probable word. Using a *beam search* across predictions instead of just an argmax could improve prediction results.

The current model used just two fixed length vectors to represent the entire input sequence. When input sequence are really long, two fixed length vectors may not be enough to capture its entire essence. Attention could help solve this issue. The technique was tried but could not be successfully implemented due to the time constraint. Adding attention to models like this one has been show to drastically improve performance of the model when the input sentences are long.

5 Transformer Model

The second model we considered was another sequence-to-sequence model that used *transformers* (See Figure 10). The introduction of transformers in 2017 and it's success in *natural language processing* tasks have caused it to replace LSTM [15]. The transformer performs better in retaining long-range context dependencies. It works on the self-attention mechanism. The recent performances of Google BERT (trained using transformers) and transformers in general motivated us to try it in our task. In this section we discuss how we developed, trained and analyses this model.

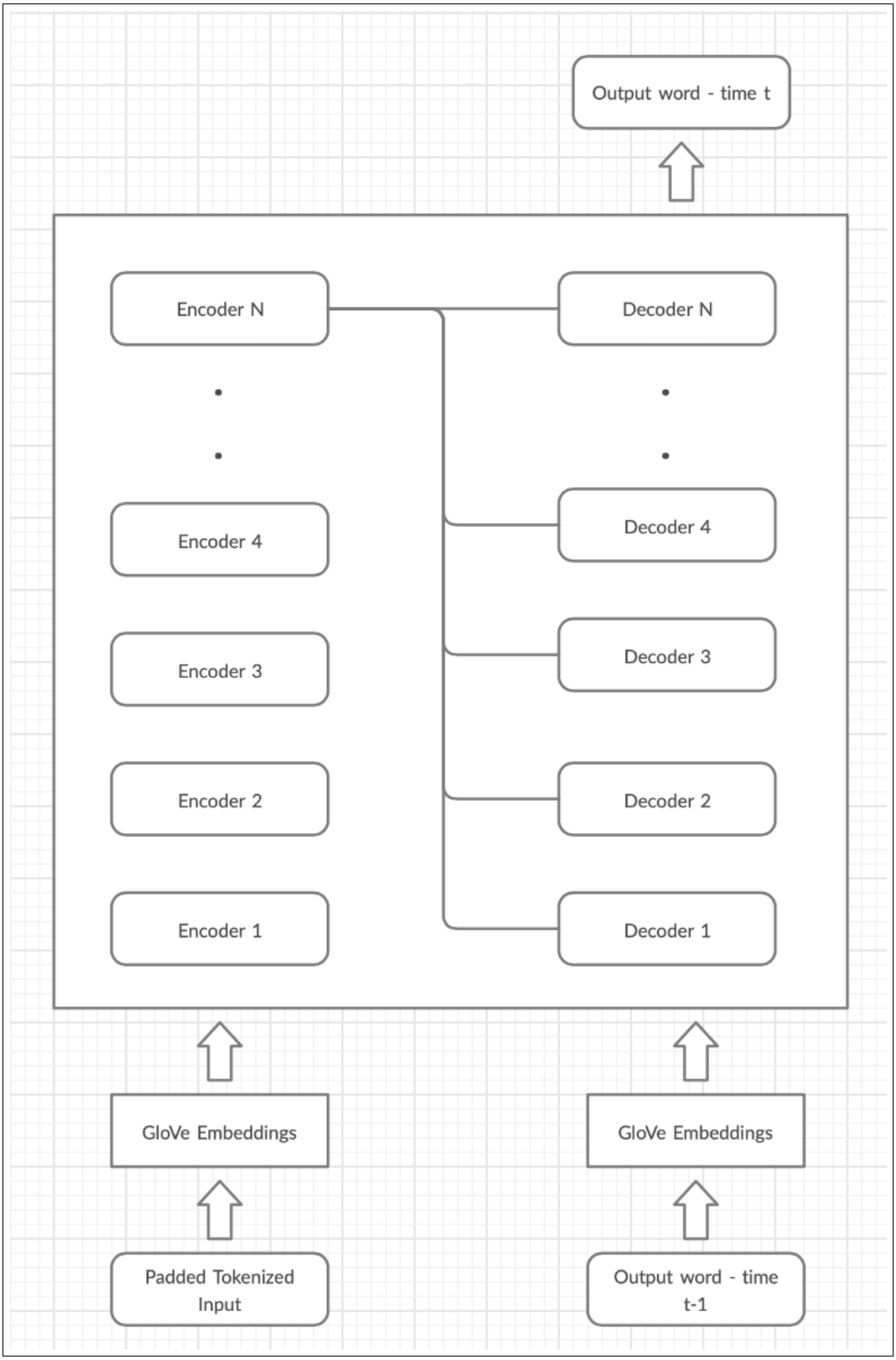

Figure 10: Transformer based Seq2Seq model.

5.1 Model Architecture

The Embedding layer uses the *subword text encoder* from the Tensorflow datasets library. The architecture as shown in Figure 11, shows the structure of a single encoder-decoder unit. The units are designed such that the outputs of each encoder is connected to every decoder. The number of such units is a hyper-parameter.

The internal structure of the encoder contains a multi-head attention mechanism and a feed-forward network (See Figure 12). The multi-head attention network computes a series of scalar dot product attentions in parallel. This produces a base for enabling self-attention and sort of acts like an ensemble technique. The self-attention is computed using three vectors - *query*, *key* and *value*. The weights for these vectors are learned during training. The dot product between each word embedding vector and each of these keys are used to compute self-attention. This process is done in parallel, multiple times. These outputs are concatenated and sent to the feed-forward network. The above process allows to jointly attend to information from different sub-spaces of representation.

The internal structure of the decoder is similar to the sub components of the encoder. The difference is the addition of a masked multi-head encoder. It is similar to the multi-head attention with a masking layer. The query vector to a decoder comes from the previous decoder layer while the key and value vectors come from the encoder network layers. The query, key and value vectors for the encoder units come from previous encoder layers. The decoder outputs from the feedfoward network is softmaxed over the vocabulary to produce the next probable word.

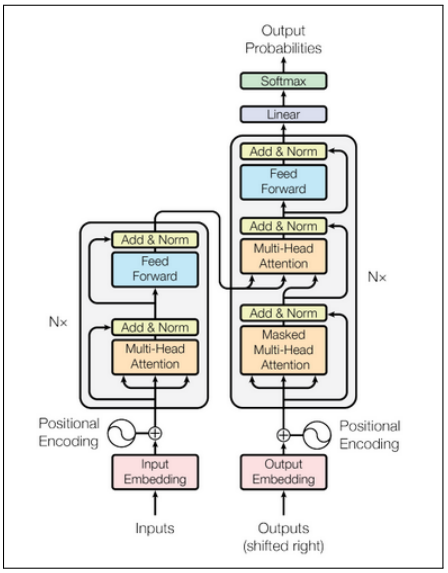

Figure 11: Encoder Decoder Unit. Image taken from [15]
.

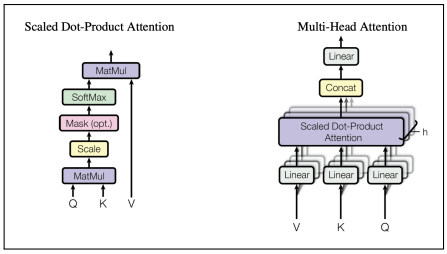

Figure 12: Attention mechanism. Image taken from [15]

5.2 Performance and Error Analysis

Dataset	BLEU Avg. Score	ROUGE-1 Score	ROUGE-ℓ	METEOR Score	WER Avg.	Human Eval.
Q&A	0	0.102	0.094	$0,035$	2.412	0.263
Joey	0.110	0.270	0.0.261	0.137	1.218	0.325

Figure 13: Table showing various scores for this sequence-to-sequence model with transformers on the Joey and generic questions and answers dataset (Q&A).

We note that the automatic metrics were run using our final trained models. That is, we ran all the experiments with a model trained first on the generic dataset and then the Joey dataset. Based on the test results, both human and automatic metrics, the transformer model works better on the Joey

dataset compared to the generic dataset. One theory as to why this happened is that the transformer model performs better on datasets with large vocabulary and the generic questions and answer dataset has a very small vocabulary. Moreover, we suspect that small size of the generic dataset compared to the Joey dataset may have been a big factor as well.

As we discussed in the previous section, considering a larger generic dataset may produce more promising results in final performance. Just like the first sequence-to-sequence model, this model consistently generated response that did not parse as proper English sentences. Future work could include trying to work on mapping how the model constructs a generic response and then tries to add 'Joey' characteristics to it. We would suggest trying to add an additional learned component that tries to implement the idea of adding characteristics. Therefore, a possible improvement to our model would be to train the model on a *part of sentence task*. This might enable the model to learn sentence formulation mappings and could structure the sentences better.

5.3 Pitfalls and Improvements

Some of the major roadblocks we faced during our implementation of this model included the construction of the transformer. There were no Keras or Tensorflow/PyTorch implementation of a basic transformer. Therefore, we had to refer to other sources to provide us with an implementations of the multi-head attention and positional encoding [16].

Another issue we faced was the inclusion/construction of context. We tried to include the entire chat history (sequence of sentences processed previously) as a context vector to each computation. However, this idea did not increase the quality of results in [15] because the compilation of the entire history/context into a single vector doesn't seem to be effective. Therefore, we worked on producing output responses independent of previous context. Other sources ([17] and [18]) indicate using the concept of goals/entities where the chatbot would recognize certain parts of the conversation as required entities (stored as context) and would work towards reaching a goal. This approach is quite specific and does not apply to general chatbots. We recognize this task as a possible area for future improvement.

6 Conclusion

In conclusion, neither model consistently generated Joey like responses (See Sections 4.3 and 5.2). We surmise that both models may understand the Joey personality but fail to generate coherent English sentences. Overcoming this issue we believe to be paramount to solving the problem if it is even possible. Furthermore, we believe that the lack of proper grammar produces the negative human evaluation results (See Figure 8 and 13).

As discussed in the previous sections, the automatic metrics are not known to be good measures of chatbot performance. Namely, it is not clear that a single saved test response would be close all appropriate Joey like responses. Hence, even a perfect model would likely fail to have good BLEU, ROUGE, METEOR or WER scores. However, our scores for both do seem slightly worse then those witness in similar projects [5, 1]. We are not aware of the specifics of the models used in this papers or the data extraction methods used. We suspect that our datasets could have been larger and better extracted to avoid incoherent statement response pairs. This would likely have produced better results for the automatic metric and the human evaluation.

Our code is available online at GitHub [19]. The final code can be found in the source folder.

Acknowledgments

We would like to thank Chenhao Tan, Brian Groenke, Aakarsh Fadnis, and Madhusudhan Aithai Mahabhaleswhwara for their feedback on this project.

References

[1] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. *CoRR*, abs/1603.06155, 2016.

[2] Suzana Ilic, Reiichiro Nakano, and Ivo Hajnal. Designing dialogue systems: A mean, grumpy, sarcastic chatbot in the browser.

[3] Qiao Qian, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. Assigning personality/identity to a chatting machine for coherent conversation generation. *CoRR*, abs/1706.02861, 2017.

[4] Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. Emotional chatting machine: Emotional conversation generation with internal and external memory. *CoRR*, abs/1704.01074, 2017.

[5] Huyen Nguyen, David Morales, and Tessera Chin. A neural chatbot with personality.

[6] O. O. Olabiyi, A. Khazane, and E. T. Mueller. A persona-based multi-turn conversation model in an adversarial learning framework. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 489–494, Dec 2018.

[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 on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[8] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[9] Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014.

[10] Nicole M. Radziwill and Morgan C. Benton. Evaluating quality of chatbots and intelligent conversational agents. *CoRR*, abs/1704.04579, 2017.

[11] Yujie Xing and Raquel Fernández. Automatic evaluation of neural personality-based chatbots. *CoRR*, abs/1810.00472, 2018.

[12] Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *CoRR*, abs/1603.08023, 2016.

[13] Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. A survey of available corpora for building data-driven dialogue systems. *CoRR*, abs/1512.05742, 2015.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[16] "a transformer chatbot tutorial with tensorflow 2.0". `https://medium.com/tensorflow/a-transformer-chatbot-tutorial-with-tensorflow-2-0-88bf59e66fe2`. Accessed: 2019-12-12.

[17] "learn designing nlp for chatbots". `https://medium.com/@phanimarupaka/nlp-design-for-chatbots-4954c2527d88`. Accessed: 2019-12-12.

[18] "chat bots — designing intents and entities for your nlp models". `https://medium.com/@brijrajsingh/chat-bots-designing-intents-and-entities-for-your-nlp-models-35c385b7730d`. Accessed: 2019-12-12.

[19] Mimic github code. `https://github.com/Nirvan66/MiMic`.