Master Thesis:

Neural response generation for email conversations

to

Prof. Dr. Stefan Lessmann

Humboldt-University of Berlin
School of Business and Economics
Institute for Information Systems

by

Sydney Richards
Matriculation Number: 592789

in partial fulfillment of the requirements
for the degree of
Master of Science in Economics

Abstract

This thesis develops a Recurrent Encoder-Decoder and a Transformer model for neural response generation in a customer support email application. The use of these methods in a closed domain allows for their evaluation using automated metrics that is otherwise unwarranted. Neural response frameworks outperform an information retrieval benchmark in the quality of responses. The Recurrent Encoder-Decoder outperforms the Transformer in both an English as well as a German data set. The automated evaluation metrics are partially correlated with human evaluations of suggested responses.

Berlin, February 25th, 2020

Acknowledgements

My gratitude goes to Andreas Rudolphi for the chance to write this thesis during my time as working student at Share Now and Prof. Dr. Stefan Lessmann for giving me the opportunity to write this thesis at his chair. Furthermore, I am very grateful for all the colleagues who participated in the evaluation survey. And finally, I am deeply indebted to my family and friends for their unwavering support.

Contents

List of Figures

List of Tables

1 Introduction

Recently, a global survey of the customer service industry showed that participants expected the use of artificial intelligence (AI) to grow by 143 percent within the next two years (Salesforce Research, 2019). At the same time 95 percent of service organizations reported to use emails as one of their primary contact channels. The most visible effect of the adoption of AI systems into customer service delivery has been the rapid spread of chatbots (Dale, 2016). Customarily, chatbots have been template-based, or in other words, the chatbot consisted of either a rule-based mechanism or a statistical matching algorithm which determined the most fitting response to a given input (Hui and Jha, 2000). Collectively, these methods are based on the concept of information retrieval (IR) and as of 2020, most solutions for generating automated responses in customer service are grounded in IR mechanisms*(Williams et al., 2015; Kannan et al., 2016). During the previous decade, the appearance of neural network based models for text generation has led to their rapid adaption in the literature for chatbot systems (Vinyals and Le, 2015; Santhanam and Shaikh, 2019). This development is driven in large part by the success of these frameworks in machine translation (MT) tasks (Sutskever et al., 2014). In principle, where template based systems ensure controllable and predictable responses, neural generation methods offer more flexible and possibly more expressive responses. Typically, the development of neural end-to-end systems involved very large data sets with millions of recorded conversations which covered a varied set of domains. Often these systems produced responses that were generic (Li et al., 2015) or lacked a consistent personality (Luan et al., 2016). In the domain of customer support through emails these issues may not be critical flaws. For example, a significant fraction of customer service communications tend to be repetitive and the conversation roles are consistent over the communication channel. In addition, a key conflict in the customer service industry is the inher-

*For example, Microsoft's *luis.ai*© or *wit.ai* a notable exception is Alphabet's recently introduced Dialogflow© which is grounded in neural methods for text generation.

ent dichotomy between the twin operational targets of keeping costs low while at the same time offering high service quality (Aksin et al., 2007). The spread of new technologies has increased the ability of employers to monitor the productivity of their employees. However, faced with conflicting employee targets in regards to quality and quantity of responses to customer enquiries, the increased use of these monitoring technologies often comes with costs to worker satisfaction and productivity (Bain et al., 2002). Conversely, as shown by Holman et al. (2002), technologies which raise worker productivity also have beneficial effects on the motivation and emotional well being of customer service agents. Nonetheless, applying deep learning methods in customer service would not only address present business concerns but also presents an interesting research case for Natural Language Generation (NLG). Specifically, as explained by Liu et al. (2016), a hot-button issue in the literature has been the evaluation of the output generated by NLG systems. Ideally, output would be evaluated manually with researchers using common criteria from the field of linguistics. Yet, considering the size of the data sets used, this is an infeasible approach and the literature has come up with numerous quantitative measures for researchers to automate the evaluation of their text generation frameworks. Even so, Liu et al. (2016) have criticized that most measures do not correlate well with human evaluations. For example, while some measures quantify the fluency of outputs through measuring the variation in output vocabulary, i.e. perplexity, others calculate the word overlap between the generated output and predetermined references. The first measure fails to address whether the output in addition to being well-behaved is actually on topic. The second group of measures is unsuitable for open-ended tasks in NLG since there are many possible responses in dialogue settings which may not be covered in the set of references used for evaluation. Conditions change in a customer service setting since the closed domain nature of subject matter limits the possible number of valid generated responses. In addition, an excessive variation of the outputs is undesirable as businesses wish to be consistent in their customer facing

communications. Consequently, word overlap measures for automated responses generated in customer support may offer a more meaningful evaluation of output quality than in open-ended dialogue settings. Furthermore, customer support often deals with a set of returning topics in conversations with customers which allows for the evaluation based on the subject matter of generated responses. Thus, this thesis will develop and evaluate the two most popular frameworks for sequence-to-sequence (seq2seq) modeling in neural text generation in the context of a language model. First, this work develops a model belonging to the class of Recurrent Encoder-Decoder (RED) models (Sutskever et al., 2014) which consists of an encoder network which step for step encodes an input sequence into a vector representation and a decoder network which generates an output sequence conditioned on the encoded vector. In order to compare their relative performance this work also uses a Transformer based neural network introduced by Vaswani et al. (2017) which contains multiple encoder and decoder layers to generate text sentences without any inherent sequential structure. The data set used for the training and testing of these frameworks consists of the email conversations in customer support for a car-sharing provider. As a consequence, the conversations will cover a certain number of topics with similar conversation structures and content which will allow for an investigation of the generation behavior of the NLG frameworks across subject matters. In summary, the model will be trained and evaluated on human generated responses in a closed domain with limited variability. Hence, automated measures of word overlap will offer a valid measure of output quality and possibly aid in investigating the interpretability of these frameworks. Next, the following section reviews the literature on NLG dialogue systems. Then, Section 3 introduces both the RED as well as the Transformer in the context of a language model. Section 4 describes the experimental design of this thesis in addition to the data used. Section 5 and 6 present the results from machine calculated, automated metrics as well as human evaluation. Finally, Section 7 concludes and discusses the contribution of this thesis to the literature.

2 Related Work

Text generation for dialogue settings as a sub-field of Natural Language Processing (NLP) has been studied long before it became computationally feasible to train deep neural networks on text data (Weizenbaum, 1966). Indeed, the applications of NLG are very diverse in and of themselves and range from areas such as machine translation of text sequences from one language to another to text summarization and even poetry generation (Santhanam and Shaikh, 2019). However, the focus of this thesis lies in language generation for dialogue systems for user interfaces. Specifically, such dialogue systems that can emulate human conversation and also can be prompted to answer questions for a specified range of topics as part of customer support conversations. Consequently, this a similar but separate approach than that of open domain systems such as chit-chat bots or conversational AI agents (Gao et al., 2018). These systems while fascinating, require data sets with millions of recorded conversations across various topics for training. Our objective is rather to see whether generative neural models for language generation in a customer service support setting can outperform the retrieval based dialogue systems which represent the current industry standard for matching pre-written templates to customer enquiries (Williams et al., 2015). Finally, the domain based nature of the data would prohibit the development of a general knowledge conversational AI system. Currently, information retrieval (IR) systems are the preferred approach used by the industry when it comes to generating text responses for human input, especially for chat and email services. Generally, IR systems are made up of a combination of matching, ranking and retrieval mechanisms for a set of predetermined responses. More specifically, this response set can either be a knowledge base of hand-written templates or a history of domain specific responses. While this approach carries a high manual workload in creating and maintaining the response set it insures that all possible responses meet requirements with regards to grammar and style (Ji et al., 2014). Matching and ranking of possible responses given a certain input can be either be

done with a rules-based or a statistical approach. On the one hand, rules-based systems require no or little data to be set up while statistical models require data for training. Popular statistical methods for IR systems in the literature have been Sector Vector Machines (SVM), term frequency and inverse document frequency (tf-idf) and more recently neural networks. Nonetheless, Józefowicz et al. (2016) found that statistical models for information retrieval tend to saturate with increasing training data, i.e. they do not make full use of the increasingly larger data sets available to researchers. In contrast, the key underlying concept of language models is to predict the probability of a word appearing in a sequence given the previous words in the sequence. Furthermore, conditional language models are a variant of language models which are conditioned on variables other than the preceding words for example personal characteristics or another text. In turn, language models only require a vocabulary and an underlying model to determine the word probabilities for each sequence. Earlier, Bengio et al. (2003) showed that a feed forward neural network was capable of modeling sequential data with a fixed vector length. Nonetheless, their approach had a serious drawback namely that their model was incapable of handling sequences of varying length. For quite some time, machine learning models belonging to the class of Recurrent Neural Networks (RNN) have achieved state-of-the art results modeling sequential data like text (Mikolov et al., 2010; Sutskever et al., 2011). Moreover, RNNs are capable of handling sequential input data by memorizing the information gained from previous sequences. Another turning point in the use of language models for text generation came with the introduction of end-to-end encoder-decoder architectures first for neural machine translation (Sutskever et al., 2014) and then for response generation (Vinyals and Le, 2015). Within this framework the encoder using a RNN converts the input sequence into a context vector. Conversely, the decoder also consisting of an RNN tries to predict a sequence using the information contained in the previous step as well as the context vector. This end-to-end method for training text generation models has

entered the literature as the sequence-to-sequence (seq2seq) model and forms the backbone of most approaches to natural language generation using deep neural networks. In addition, researchers have experimented with using convolutional neural networks for both encoding and decoding sequential data. As shown by Kalchbrenner et al. (2016), convolutions offer the benefit of parallelizing over the sequences making training much quicker in general. Regardless, the majority of work done on neural text generation using encoder-decoder frameworks has been done using recurrent structures for both encoding and decoding. It is important to note that these initial seq2seq models had issues encoding longer input sequences into a fixed length context vector. Thus, the decoder would loose information from the inputs if sequences were sufficiently long. Solutions to the issue were proposed by both Bahdanau et al. (2015) and Luong et al. (2015) called attention which improved the quality of neural machine translation and text generation systems alike. Attention allows the model to focus on the relevant steps of the input sequence as needed and has become a fixture in the text generation literature. Similarly, the most recent neural text generation architectures do away with recurrence and convolutions entirely and are based solely on attention mechanisms. The Transformer model introduced by Vaswani et al. (2017) consists of stacks of six encoders and decoders respectively. Moreover, each encoder and decoder consists of a self-attention module as well as a feed forward neural network and has been proven to require significantly less training time than models with similar performance which are based on recurrence. Consequently, Table 1 gives a partial overview of the research on neural response generation in chronological order. The table shows which data was used in the development of the model as well as the evaluation metrics used. Likewise, the table denotes which model framework was used and the specification that was introduced in the paper. As is clearly visible, the majority of text corpora used was quite large as the intent of building a open-domain conversational AI would suggest. Most related to the approach suggested in this thesis for generating responses for email conversations

in customer service settings is the work of Kannan et al. (2016) and Henderson et al. (2017). In both cases the authors use a RNN encoder to convert incoming email messages into a vector representation. However, their approach differs from true end-to-end models in that they use this context vector in order to retrieve the most fitting reply from a permitted response set. Basically, their approach is grounded in the IR principle introduced earlier, while this work aims to propose an end-to-end text generation framework. They argue that this ensures that the system will only generate high quality responses. This thesis argues that in case of customer service support, the training data will mostly include high quality responses. Therefore, the response set determination is skipped and an end-to-end framework for response generation can be used. Meanwhile, the literature shows there are significant obstacles when training end-to-end frameworks for neural response generation. In practice, plain vanilla neural generation models tend to produce simple or obvious responses such as "*I don't know*" (Serban et al., 2015; Vinyals and Le, 2015) which makes them useless for generating interesting and coherent responses. Li et al. (2015) argue that optimizing for the likelihood of outputs given a set of inputs leads to a tendency to produce commonplace responses. Furthermore, they argue this phenomenon is caused by the relative frequency of generic responses compared to the relative scarcity of more specific responses. In turn, they propose an objective function which not only optimizes for the likelihood of responses given inputs but conversely for the likelihood of inputs given a response. They capture this intuition by using the Maximum Mutual Information (MMI) criterion when decoding output sequences during inference. Alternatively, Li et al. (2017) cast the task of producing dialogue responses as a reinforcement learning problem and train a generator as well as a discriminator. Specifically, they propose using a generative adversarial neural network in order to discriminate between messages generated by humans or by the machine. The discriminator punishes the system for producing outputs which are identified as machine-generated leading to more content-full and less generic responses. An-

other research field in neural generation tasks has been the concept of embedding personas into dialogue response systems for the purpose of generating more empathetic responses. For example, Luan et al. (2016) propose that speaker roles inform phrase patterns, authority claims and naturally different speaker roles are associated with different speech markers. While the questioner will use words like how and why, a responder will use more directive words. Similarly, Li et al. (2016) introduce a speaker-addressee model which uses an embedding vector to capture speaker- and addressee-specific attributes among others age, gender and dialect which influence content and style of model responses. Intuitively, speech patterns are not only influenced by the the traits of the speaker but also by the traits of the addressed individual. Another research branch has been concerned with the concept of memory networks introduced by (Weston et al., 2015). They showed that by storing the input sequences in a memory vector which can be used for training, RNNs may be able to attend to information over longer distances. In principle, when training end-to-end neural model for text generation memory can work in a similar fashion than the attention mechanism. To illustrate, memory networks store entire input sequences in physical memory for later reference whereas attention uses fixed length vector representations. However, Sukhbaatar et al. (2015) showed that memory is especially useful for multi-turn conversations as it allows to store information across conversation pairs. Conversely, memory networks are demanding with regards to computation power and offer almost no gains over attention for single turn conversations. Yet, we do not aim to model multi-turn conversations but rather responses to single-turn customer support conversations. As such, a attention mechanism is preferable due to its lower hardware requirements. Lately, Transformer based models have become ubiquitous in the field and Adiwardana et al. (2020) have shown in their recent study that transformer based architectures can outperform existing systems by considerable margins. Thus, this thesis compares the Transformer and RED architectures for neural text generation to an IR benchmark.

Table 1
Literature Review: Neural Response Generation

Publications listed in chronological order							
Study	Dataset		Model Architecture		Training		Evaluation
	Corpora	Size	Base	Variation	Loss Function	Decoder	Metrics
Vinyals and Le (2015)	IT-Helpdesk OpenSubtitles	3M 62M	RED	Vanilla	Cross Entropy	greedy	Human Eval
Li et al. (2015)	Twitter OpenSubtitles	23M 62M	RED	Attention	MMI	MMI modified Beam Search	BLEU Human Eval
Serban et al. (2015)	Movie Triples	200K	RED	HRED	Cross Entropy	Beam Search	Perplexity
Serban et al. (2016)	Ubuntu Dialogue	500K	RED	VHRED	Cross Entropy	Beam Search	Human Eval
Li et al. (2016)	TwitterPersona IMSDb	25M 70K	RED	Persona	MMI	MMI modified Beam Search	BLEU
Kannan et al. (2016)	EMail messages	238M	LSTM Encoder	IR	Logistic Loss	-	MRR Precision@K
Li et al. (2017)	OpenSubtitles	62M	RED	GAN	Cross Entropy	MMI modified Beam Search	AdverSuc Human Eval
Henderson et al. (2017)	EMail Messages	300M	Feed Forward	IR	Multi-Loss	-	Precision@K
Rashkin et al. (2018)	Empthatic Dialogues	30K	Transformer	Empathy	Cross Entropy	Beam Search	BLEU Perplexity
Wolf et al. (2019)	Persona-Chat	700M	Transformer	Transfer	modified Cross Entropy	Beam Search	Perplexity Hits@1
Adiwardana et al. (2020)	Various Social Media	867M	Transformer NAS	Evolved	modified Cross Entropy	Beam Search	Perplexity Human Eval

List of abbreviations used: BLEU - bilingual evaluation understudy, GAN - Generative Adversarial Network, HRED - Hierarchical Recurrent Enocder-Decoder , ISMDb - Internet Movie Script Database, IR - Information Retrieval,NAS - Neural Architecture Search, MRR - Mean Recipocal Rank, MMI - Maximum Mutual Information, VHRED - Latent Variable Hierarchical Recurrent Encoder-decoder

3 Neural Text Generation

Before discussing in detail the concepts of the approaches considered in this thesis, we expand more formally on the setting of text generation using a language model. At its core a language model simply describes a probability distribution over a sequence of words. For example, consider a sequence $Y = (y_1, y_2, ..., y_T)$ of length T then using the chain rule we can establish the following relationship

$$p(Y) = \prod_{t=1}^{T} p(y_t \mid y_1, ..., y_{t-1}) \tag{1}$$

for a word at step t given the preceding word tokens already observed. Basically, neural response models attempt to map input sequences $S = (s_1, s_2, ..., s_J)$ to an output sequence Y. Thus, we can generalize by conditioning the word probabilities on the input sequence S which gives

$$p(Y \mid X) = \prod_{t=1}^{T} p(y_t \mid S, y_1, ..., y_{t-1}) \tag{2}$$

In our specific application we aim to train models which reliably generate readable and topical replies Y to customer enquiries S. In the following, we restrict ourselves to approaches which have proven themselves for language modeling. In principle, neural language models parameterize this problem by one-hot encoding the previous word of a sequence into a feature vector $x \in \mathbb{R}^d$ that uses d different features to describe the context. For example, let us assume that each word in our Vocabulary is associated with a Word ID k then one-hot encoding returns a vector where only the kth item equals 1 whereas the rest is zero. Next, we seek to determine the probabilities over our vocabulary V using a weight matrix $W \in \mathbb{R}^{|V| \times N}$ which describes the relationship between feature values and scores for each word in our vocabulary and a bias vector $b \in \mathbb{R}^{|V|}$. Thus, we could

calculate probabilities over our vocabulary using a simple multi-layer perceptron:

$$\begin{aligned} h_t &= g(W_{xh}x_t + b_h) \\ \widehat{y}_t &= softmax(W_{yh}h_t + b_y) \end{aligned} \tag{3}$$

where computation is split into two stages. First, the hidden layer takes an the feature vector x as input and returns a hidden vector h where function $g(\cdot)$ denotes a non-linear activation function such as tanh. Second, the output layer which takes the hidden inputs h and transforms them into probabilities $\widehat{y}$ using a softmax activation function. Until now this model is only able to capture a fixed length context depending how the feature vector x is constructed. For example, models which are used to generate basic word embeddings, such as continuous bag of words based word2vec (Mikolov et al., 2013), use a fixed window around the target word in order to train an embedding matrix W. Conversely, RNNs are a variety of neural networks which are capable of flexibly modeling long-distance dependencies by connecting hidden states across steps(Elman, 1990). Specifically, when calculating the hidden state h_t RNNs make reference to the hidden state at $t-1$:

$$h_t = \begin{cases} g(W_{xh}x_t + W_{hh}h_{t-1} + b_h) & t \geq 1, \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Although appealing in theory, RNNs as described above are rarely applied in practice since the problem of vanishing or exploding gradients prevents the model from learning long sequences. In particular, as explained by Bengio et al. (1994) the problem stems from propagating local errors at each time index. Hochreiter and Schmidhuber (1997) addressed this issue by introducing LSTM which include self-connected "gates" which allow RNN units to manipulate the hidden sate and regulate the flow of information to and from memory. Another computationally more efficient solution to the problem of vanishing gradients in recurrent networks was offered by Cho et al. (2014). Namely, they introduced the gated recurrent unit (GRU) which uses two gates that are represented by the following set of

equations:

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \tag{5}$$

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{6}$$

$$\tilde{h_t} = tanh(W_{xh}x_t + W_{hh}(h_t \odot h_{t-1}) + b_h) \tag{7}$$

$$h_t = (1 - z_t)h_{t-1} + z_t\tilde{h_t} \tag{8}$$

the first equation denotes the reset gate r_t which decides how much information from the previous hidden state h_{t-1} is carried over. Furthermore, z_t denotes the update gate which selects whether the hidden state is updated with the new candidate hidden state $\tilde{h_t}$. This allows the hidden state to "forget" any information that is found to be irrelevant later in the future. Next, we will embed the GRU concept into the encoder-decoder framework which has been a popular choice for neural text generation.

3.1 Recurrent Encoder-Decoder with Attention

At a very high level a RED model first reads an input sequence using an encoder to construct a "thought" vector of real-valued numbers with fixed length. Consequently, this vector captures the meaning of the input which is used by the decoder to produce an output sequence. In this application both the encoder and decoder are made up of neural networks containing GRU layers. In the encoder network after passing through an embedding layer the sequences are read in reversed order by the first GRU layer. Conversely, the second GRU layer reads the sequences first to last. We hope that by calculating hidden states in both directions that we can encode summaries for both the preceding and the following words for each step in the sequence. Next, the final hidden state of the first neural network is used to initialize the hidden state of the second neural network in order to generate the target sequence. In a base specification the encoder compresses the available information of the input sequence into a fixed length vector

from which the decoder generates the appropriate reply. Both Bahdanau et al. (2015) and Luong et al. (2015) have pointed out that this may make it difficult for the neural network to cope with longer input sentences. In turn, both groups of researchers have proposed an attention mechanism which assigns a weight to the sequences of hidden states returned by the encoder which the decoder uses to predict the next word in the output sequence. In addition, these attention weights have the added benefit of allowing us to open the black box and let us see which parts of the input sequence where considered important for predicting the output sequences. We express the encoder as $GRU^{(e)}(\cdot)$ and the decoder as $GRU^{(d)}(\cdot)$ respectively where the encoder takes $S = (s_1, s_2, ..., s_J)$ as an input and the decoder generates sequence $Y = (y_1, y_2, ..., y_T)$.

$$h_j^{(e)} = \begin{cases} GRU^{(e)}(s_j, h_{j-1}^{(e)}) & j \geq 1, \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$a_{tj} = h_{t-1}^{(d)} h_j^{(e)T} \tag{10}$$

$$\alpha_{tj} = \frac{\exp(a_{tj})}{\sum_{j=1}^{J} \exp(a_{tj})} \tag{11}$$

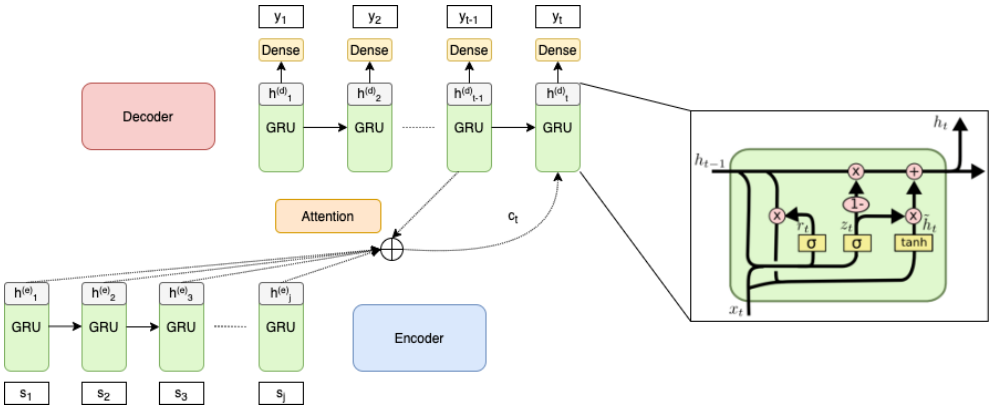
$$c_t = \sum_{j=1}^{J} \alpha_{tj} h_j^{(e)} \tag{12}$$

$$h_t^{(d)} = \begin{cases} GRU^{(d)}(y_{t-1}, h_{t-1}^{(d)}, c_t) & t \geq 1, \\ h_J^{(e)} & \text{otherwise} \end{cases} \tag{13}$$

In the encoder phase the model calculates the encoder hidden state $h_j^{(e)}$ for the j-th word of the sequence S. The first hidden state vector is initialized with 0 i.e. $h_0^{(e)} = 0$ and the final hidden state $h_J^{(e)}$ has seen all the words in the input sequence. Theoretically, the final hidden state should be able to encode all of the information in the source sentence. However, as before one may worry that encoding the information of the input sequence into a hidden vector of arbitrary length might be an inadequate representation of the available information. The attention mechanism introduced by Bahdanau et al. (2015) is a natural way

Figure 1

Recurrent Encoder-Decoder Framework with Attention



The graph is a schematic representation of the Recurrent-Encoder Framework used within this thesis. For simplicity, only a single encoder layer is depicted.

for solving the transmission problem from encoder to decoder layer. Basically, as opposed to decoding from a single vector representation, attention references the hidden state of each step in the input sequence. This is done by calculating attention scores a_{tj} at each step t during decoding. Specifically, scores are used to measure how well inputs at position j matches with an output at position t. Bahdanau et al. (2015) calculate these scores through a feed-forward neural network which is trained jointly with all other components of the framework. However, this approach introduces additional parameters which need to be trained therefore increasing the computational resources required. In contrast, the dot attention introduced by Luong et al. (2015) as given by equation (10) relies solely on matrix multiplication and has shown similar performance. Next, using a softmax activation function these attention scores are translated to probabilities α_{tj} which measure the probability that output word y_t is aligned with source word s_j. Moreover, a context vector c_t at each step t is constructed by summing over the j hidden state vectors of the encoder weighted by α_{tj}. Lastly, at each decoding step t the decoder hidden state $h_t^{(d)}$ is concatenated with the context vector c_t and run through a final feed-forward layer which returns the vector of word probabilities y_t for the target vocabulary. In the following section, we present the network architecture introduced by Vaswani et al. (2017).

3.2 The Transformer

In principal, the Transformer follows the general encoder-decoder architecture but does away with recurrence and convolutions and relies solely on an attention mechanism to draw long-term dependencies between the input and output sequences. Likewise, as the model has no inherent sequential structure the authors use a positional encoding for the embeddings to initialize both output and input sequences. These positional encodings have the same dimension as the word embeddings such that the two can be added together. Typically, word embeddings are d-dimensional vectors which represent how close two words are in their meaning. Additionally, by adding the positional encodings to the word embeddings the vectors represent how close together words are in their meaning and position. The attention mechanism used in the Transformer is close in concept to the dot-product attention mechanism described in Luong et al. (2015) but adds a scaling factor to prevent attention scores from becoming to large. Specifically, the result of the matrix multiplication is scaled by depth of the second operand. Vaswani et al. (2017) define their underlying mechanism for obtaining their attention weights as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_K}})V \tag{14}$$

where query Q, key K and values V are vectors which are generalized concepts and useful when thinking about attention. For example, in the previous section the attention mechanism in the decoder layer used the current decoder hidden state as the query vector and the encoder hidden states as key and value vectors respectively. Additionally, we define the sequence of word inputs as a matrix $X \in \mathbb{R}^{N \times d}$ where N represents the length of sequences. Moreover, Q,K and V are obtained through matrix multiplication of the input X with the trainable weight matrices $W^Q \in \mathbb{R}^{d \times d_Q}$, $W^K \in \mathbb{R}^{d \times d_K}$ and $W^V \in \mathbb{R}^{d \times d_V}$ respectively. As shown in equation (14) Vaswani et al. (2017) compute the dot product of the

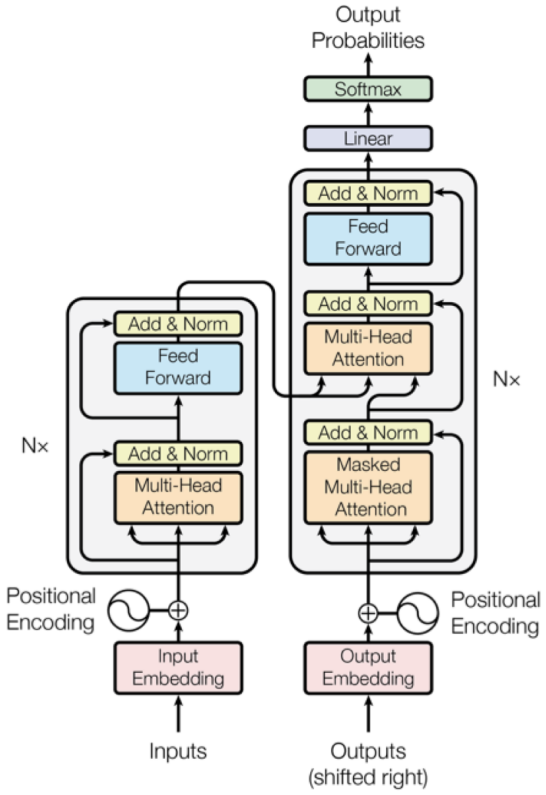
query with all keys to obtain the attention scores which measure how much to focus on other sections of the input sequence when encoding each step. Next, they scale the product with the square root of the depth of the key vector d_K for the purpose of preventing very large gradients for large values of d_k. Also, they apply a softmax function onto the scores in order to obtain the normalized attention weights which add up to 1. Lastly, the attention outputs are obtained by multiplying the attention weights with the value vector. In principle, the output of the scaled-product attention should contain information on the current position from all other encodings of the sequence. Neverthless, the information on the current position might dominate and as a consequence Vaswani et al. (2017) propose splitting Q, K and V across h multiple heads in order to jointly attend to information at different positions where:

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, ..., head_h)W^O \\ where\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{15}$$

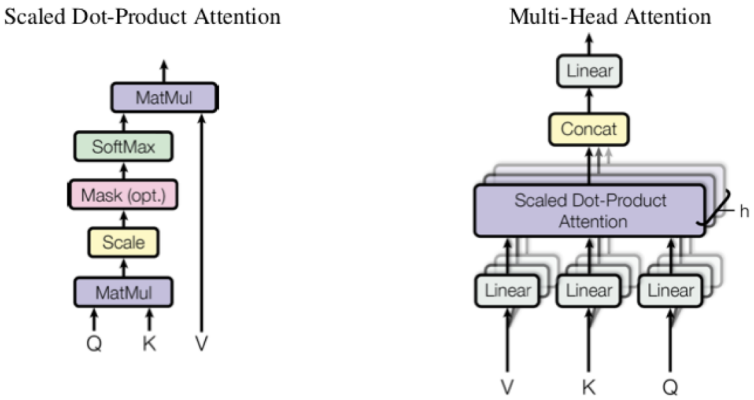
In addition, multi-head attention initializes and trains separate weight matrices W^Q, W^K and W^V for each head which allows the attention mechanism to train using different representational subspaces. For simplicity we assume that the weight matrices are of depth $d_k = d_v = \frac{d}{h}$. The separate attention outputs are then concatenated and finally run through a feed forward network consisting of two fully connected layers with rectified linear unit (ReLU) activation between them. In total, the transformer uses multi-head attention in three separate ways. First, the encoder layers self attends to all positions in the input sequences at each step during encoding. Furthermore, the decoder layer incorporates two forms of attention. Second, similarly to the RED model, the encoder-decoder attention layers use queries from the previous decoder layer as well as key-value pairs from the encoder layer to allow each position in the decoder sequence to attend to all positions from the input sequence. Lastly, the decoder also contains self-attention which allows each position in the decoder to attend to all positions from

Figure 2
The Transformer

(a) *Model Architecture*



(b) *Attention Mechanism*



Source: Vaswani et al. (2017)

the previous decoding layer leading up to and including that position. For the model to attain the autoregressive properties of the language model the authors implement a forward looking mask which prevents the model to connect tokens that are further along in the sequence with the current position. Lastly, the Transformer stacks encoder and decoder layers respectively were each decoder layer is connected to all encoder layers. For now we have only discussed methods for estimating word probabilities in the context of a conditional language model. While this may suffice for training, the neural network has not yet generated any actual text. Thus, we turn to methods for decoding these probability distributions into word sequences during decoding for inference.

3.3 Decoding

In general, during inference on a test set the trained RED or Transformer models respectively will be fed the input sequence and generate an output. Both frameworks take two inputs in particular, the input sequence and a placeholder output sequence containing only a start token which indicates the beginning of a sequence to the model. The model will return the attention weights as well as a vector of word probabilities over the target vocabulary for each word in the target sequence. In the scope of this thesis, input sequences will consist of customer enquiries and the target sequence is the appropriate response from customer support. The most straightforward approach would be to take the word with the highest probability, concatenate it with the start token and together with the original input sequence feed it back to the model to generate the next word in the target sequence. Ideally, this process is repeated until the model generates an end token which indicates that the model considers the response to be complete or we reach a pre-defined threshold for sequence length. This iterative search for the first best solution at each step t is called *greedy search*. However, it is important to note that this approach does not always lead to the model decoding the response sequence with the highest probability. To illustrate,

imagine a scenario in which maximizing the probability for the first word prevents the model from generating the next word which would give the overall sequence a higher probability. Instead of only extending the sequence with the first best solution, one could maintain multiple hypothesis by taking the b-best solutions at each iteration. Commonly, this approach is referred to as *beam search* (Sutskever et al., 2014) wherein at each step b-hypothesis are extended with b-words with the highest probability. This approach while being more costly computationally, usually leads to sequences with higher probabilities with typical values ranging from b between 2 and 8 (Huang et al., 2018). During the model development in this thesis, beam search decoding offered no gains in output quality but was very expensive computationally. As such, all models in this thesis use a greedy search decoding algorithm. Next, we will describe the domain for the previously introduced neural response model architectures and describe the data which will be used to train these networks.

4 Experimental Design

Until now, the task of generating automated email responses for customer services support required a deep knowledge of the business domain and relies heavily on manual labor for the construction of business rules and or response sets. In contrast, the previously introduced neural architectures are able to be built with limited knowledge of business processes, given that the data set contains the necessary responses towards customer enquiries. In turn, we will use the email conversations in customer support of a free-floating car sharing service provider in order to assess the quality of the proposed architectures. Likewise, the data set contains the responses of actual customer service agent towards customer enquiries as such we have only quality responses since agents undergo frequent quality reviews and trainings. Consequently, by training these frameworks to behave like real life customer service agents, we can replicate these quality responses thereby lowering the workload of the customer support team. Ideally, in order

prove that the experiment carries external validity for other business domains or settings we would transfer the proposed architectures to other data sets in other business domains where similarly high quality responses are available. As this was not possible, we will use the data sets of two separate markets, with different languages namely North America and Germany to see whether these frameworks can be generalized. In the following, we describe the data and processing methods used as well as illustrating the business setting.

4.1 Data

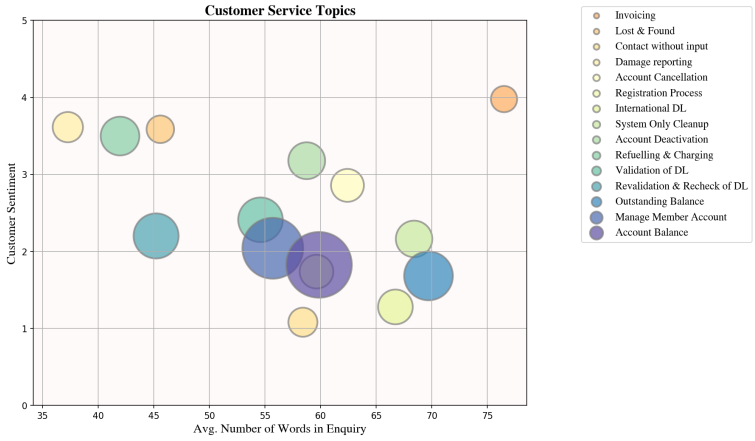
For North America 149.492 email conversations from January 2018 until December 2019 were gathered. The German data set covers 148.898 email conversations in 2019. Conversation pairs of customer enquiries and customer support responses are constructed by matching email activities which contain the email body and further meta data through an assigned common case number. It is important to consider that the customer is able to write emails from within the company's app service, as well as over her personal email account. Consequently, the input form varies greatly as some messages are written in a formal manner and others may just contain a few keywords. Furthermore, most emails can be classified into certain topic categories for which the customer enquiries and support responses are often similar. For example, customer communications regarding lost personal items (e.g. "Lost & Found") usually are longer due to a higher case complexity. On the other hand, the company offers its customers credits if they refuel a rented vehicle. This process requires that the client sends a copy of the fuel receipt and she will receive credits towards her company account. Moreover, clients can rate their interactions with the company's customer support service and unsurprisingly refueling rebates are rated better on a Likert scale from one to five than communications due to accounting issues. The first part of Figure 3 shows this relationship between message topic, case complexity and customer sentiment. Message complexity is approximated with the length of the incoming

message and use a post-contact customer rating as a measure for sentiment. In

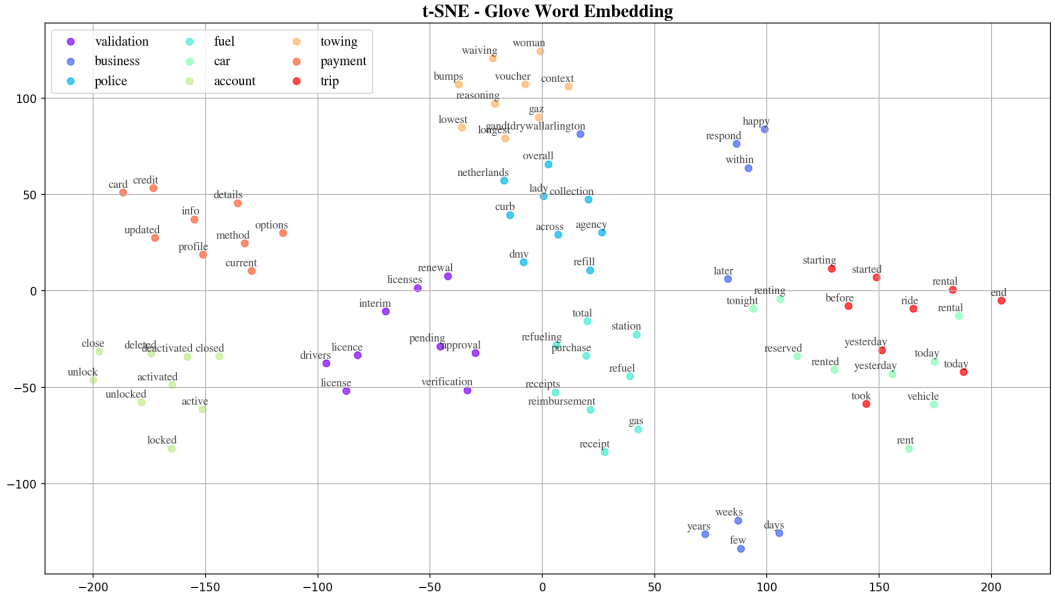
Figure 3

Visualizing Customer Support Communications

(a) *Descriptive Statistics*



(b) *Word Embeddings*



(a) The scatter plot shows the average message length of a customer enquiry as well as the customer rating of the customer support service on a Likert Scale. The plot size shows the relative share of the topic of the overall message volume.
(b) In order to obtain an idea of the trained word embeddings, the 9 most similar words are depicted for a group of keys which are strongly related to the prevalent customer service topics.

addition, the second part of Figure 3 shows nine frequent keywords as well as their most similar word vectors from a GloVe word embedding trained on a corpus of incoming messages (Pennington et al., 2014). Ideally, this word embedding is able to relate the message content to the preeminent topics which should increase the performance of the encoder frameworks. A t-Distributed Stochastic Neighbor

Embedding (t-SNE) is used in order to obtain a reduced representation of the d-dimensional word embeddings (see van der Maaten and Hinton (2008)) such that they can be projected on a two dimensional scatter plot. It is important to note that the axis scales contain only information on the relative closeness of words within a word cluster but give no indication of absolute relationships between words or word groups. The fact that we can identify word clusters which are related to the keywords indicates that the trained embedding vectors are able to reflect the business domain.

4.2 Data Processing

Now we turn to a description of the data pipeline which is used to prepare the raw email data for training. The company server stores incoming and outgoing messages in a raw html format as such a parser is used to extract the text content. Since many messages are replies or re-replies, message bodies are split so that they only contain the current thread. Also, all email headers and footers are removed, so that the models can focus on meaningful content. Unlike the data processing pipelines for many text classification tasks, no lemmatization or stemming on the word tokens is performed. Above all, during decoding, when the models output text, it should be as close to natural text as possible. Thus, using stemming would prevent the model from analyzing text in its natural form. Next, tokens are used to replace names as well as money amounts. The text sequences are then stripped of all non-alphanumeric characters except for punctuation. A $< start >$ and $< end >$ token is added to every sequence allowing the models to identify the beginning and ending of sequences during decoding. Lastly, to reduce the computational load each sequence is limited to fifty word tokens and padded with zeros if shorter. In the following, incoming customer messages are described as the inputs and the customer service responses as the targets. Here it is important to mention that two different tokenizers are used for the inputs and the targets to convert the text sequences into integer numbers and vice versa. This is done

to ensure that some words which may appear in the inputs do not appear in the generated target sequences. Thus, ensuring politeness and appropriateness in the vocabulary used for the generated responses.

4.3 Benchmark Model

In order to assess the relative performance of our neural response models an information retrieval system with a set of predefined responses is implemented. The response set contains message templates which are to be used by newer customer service agents since they provide institutional knowledge when responding to customer enquiries. In similar fashion to an industry solution proposed by Williams et al. (2015), a tf-idf model is trained for similarity between input sequences and templates in order to retrieve the appropriate template given an input sequence. The tf-idf measure consists of two components. The first measures the frequency of individual tokens within a sequence. Second, the document frequency counts how often a token appears within a corpus (Sparck Jones, 1988). To obtain the tf-idf measure, term frequency is then multiplied with the inverse of the document frequency. In this use case for example, words like "payment" will have a high tf-idf value and will be representative of input sequences in the "Account Balance" topic. Consequently, the model will match a template with a similiar tf-idf value for "payment" and the other word tokens in the input sequence. Alternatively, one may have applied a neural network in order to solve the classification problem similarily to Kannan et al. (2016). However, the tf-idf measure has been a proven benchmark for information retrieval tasks and is comparatively efficient to compute (Robertson, 2004).

4.4 Model Training

Both the reference model as well as the two neural frameworks are implemented using Python on Google Colab notebooks running on a hosted runtime with GPU accelerators. Moreover, the neural networks are implemented using the

TensorFlow v2.1 library which is designed to take full advantage of the GPU accelerators thereby lowering training times significantly. First, our implementation of the RED model is very similar to the set up shown in Figure 1. Following Bahdanau et al. (2015), we use two GRU layers in the encoder where the first layer reads sequences in the order they appear whereas the second GRU layer reads input sequences in the reverse order. Intuitively, this should ensure that our encoder hidden states h_j contain information from both the preceding as well as the following words. Both encoder layers implement recurrent dropout in order to prevent overfitting on the training data. Moreover, all layer weights use a glorot uniform initializer. Unlike Bahdanau et al. (2015), we do not use a FFNN in order to calculate attention scores but rather follow the approach taken by Luong et al. (2015) and calculate the dot product of the encoder output and the decoder hidden state vector at each step in the target sequence. The output of the GRU layer in the decoder is then run through a linear unit which at each step of the generation of the output sequences returns a vector of word probabilities over the target vocabulary. The meta-parameters for our Transformer model follow the ones chosen in Vaswani et al. (2017) and the meta-parameters for both frameworks are shown below in Table 2 .

Table 2

Meta-Parameters for Neural Generation Models

	Recurrent Encoder-Decoder	Transformer
Learning Rate	0.001	Custom Schedule
Loss Function	Categorical Cross Entropy	
Optimizer	Adam	
Embedding Size	256	
Batch Size	64	
Hidden Units	1024	
Dropout Rates	0.2	0.1
No. of Encoder Layers	2	6
No. of Decoder Layers	1	6
No. of Attention Heads	-	8
No. of Training Steps	11,240	8,992
No. of Parameters	97,822,046	69,076,575

Custom Schedule:$lrate = d_{model}^{-0.5} * min(step\_num^{-0.5}, step\_num * warmup\_steps^{-1.5})$

4.5 Evaluation Metrics

Due to the discrete nature of the data a big issue in the development of natural language generation systems has been the question of how to assess the quality of the generated text sequences. Ideally, we could draw on a large number of customer service professionals which could manually evaluate the output of our models. Unfortunately, this is not an option. Therefore, we must draw on the growing evaluation literature for NLP applications. From the Review Table 1 we can see that the bilingual evaluation understudy (BLEU) introduced by Papineni et al. (2002) has been among the most popular and therefore most used quantitative evaluation measures. Originally developed in order to assess the quality of machine translations, BLEU is a widely adopted precision-based measure used for evaluating machine generated translations, summaries and as in this case dialogue responses. The idea behind the measure is to count how often the n-grams of the generated response appear in a reference response and divide that number by the total number of n-grams in the candidate response. It is important to clip the count of n-grams in the generated candidate response by the maximum count of the specific n-gram in the reference response. Otherwise, one could inflate BLEU scores by simple n-gram repetition. In our application the reference response will be the actual reply of a human customer service agent to the customer enquiry. Furthermore, by averaging BLEU scores for unigram, bigram, trigram and quad-gram precision we can capture both the adequacy (through unigrams) and fluency (through longer n-grams) of our responses. Finally, the score contains a brevity penalty which punishes candidate responses that have many matching n-grams with the reference but are much shorter in total. Nonetheless, the use of BLEU has been criticized in the literature (Novikova et al., 2017; Reiter, 2018) for not correlating with human evaluations of model output. Consequently, we consider ROUGE (Lin, 2004) a modification of BLEU which focuses on recall by dividing the n-gram count of the reference through the n-gram count of the candidate response. Additionally, we consider METEOR (Banerjee and Lavie, 2005)

a modification of BLEU which considers both recall and precision. More specifically, the unigram implementation of ROUGE termed ROUGE-1 is used which checks for matches on the word level between reference and candidate responses. Naturally, one would expect values for METEOR metric to fall between values for BLEU and ROUGE scores. However, the different implementation packages for Python use different brevity penalties and in addition the BLEU and METEOR metrics use different number of n-grams in their implementation making comparisons across metrics difficult. Also, It is important to note that the word overlap measures fail to consider the use of synonyms. Hence, low score values may actually not capture divergence between candidate and reference. Nonetheless, the repetitive nature of customer support conversations should mitigate this.

5 Results

First, the mean results for the quantitative measures for the test data set are reported in Table 3. All three indices are measured on a scale from 0 to 100 and a higher value aligns with a higher quality response in the sense that it aligns more closely with the human generated gold standard reference. For both the English data set as well as the German data set, the results show that the both neural models outperform the baseline information retrieval model in all three categories by significant margins. Furthermore, the RED model outperforms the Transformer based model where the difference in performance seems to be greater in the English data set compared to the German data. The similar performance of all three frameworks across different languages indicates that this approach evaluates their actual value for natural language generation and not their performance on a certain data set. The performance of all three models is also compared across the five most popular topics in the data set in order to help explain their behavior. One might assume that these neural frameworks perform better in areas where conversations are more schematic and variate less. Hence, the models should have an easier time recognizing and matching the patterns in

Table 3
Results: Automated Evaluation Metrics

	English Data			German Data		
Model	BLEU	ROUGE-1	METEOR	BLEU	ROUGE-1	METEOR
Baseline	3.48	21.51	9.51	3.19	15.83	8.66
RED	**14.84**	**33.30**	**30.26**	**14.73**	**32.42**	**33.64**
Transformer	7.98	26.05	19.58	12.43	27.24	23.19

BLEU: $= min(1 - \frac{r}{c}, 0) + \sum_{n=1}^{N} \frac{1}{N} log(p_n)$
ROUGE-1:$= min(1 - \frac{r}{c}, 0) + \ log(re_1)$
METEOR: $= min(1 - \frac{r}{c}, 0) + \ \sum_{n=1}^{N} \frac{1}{N} log(\frac{10 \cdot re_n \cdot p_n}{re_n + 9p_n})$
where r: reference length, c: candidate length, p_n: n-gram precision, re_n: n-gram recall
All evaluation metrics are referenced with an actual human response.

the human generated responses. By looking at the averages for our metrics across a selection of popular topics in Table 4, we can see a large variation in the metric values for our English data set. For example, values for the BLEU metric bottom out at 1.28 for the Transformer model. This indicates a very poor match of the created response and the reference response. In fact, as mentioned by Xie (2017), this may be caused by repeated outputs during decoding. Later, we investigate more closely the decoding behavior of the two neural models using the attention weights. Nonetheless, the performance of both the RED as well as the Transformer model seems to be correlated across topics. For example, while the two neural models both achieve high scores in "Account Balance" and "Outstanding Balance" both receive lower scores in the areas "Manage Member Account" and "Validation of DL". Meanwhile, the variation of the performance measures across topics is lower in the German data set, hinting at a much more stable performance across various enquiry types. Moreover, it is noteworthy that the difference in performance between the RED and the Transformer model seems much smaller than in the English data set. Recall the generic response criticism of Li et al. (2015) which stated that many neural response models produce rather generic or bland output. As a consequence, it is possible that the two neural models only produce valid outputs for topics which lend themselves to generic output. This is supported by the fact that the models perform well in the "Outstanding Balance" topic in the English data set which can be defined as common and monotonous.

Table 4

Results: Metrics Breakdown by Topic

	English Data			German Data		
Topic	BLEU	ROUGE-1	METEOR	BLEU	ROUGE-1	METEOR
Account Balance						
Baseline	7.55	25.64	9.46	2.16	14.15	8.66
RED	17.42	37.65	**40.01**	**9.13**	**24.71**	**26.12**
Transformer	**19.79**	**37.74**	32.61	7.59	21.64	18.24
Manage Member Account						
Baseline	3.35	20.98	9.51	2.43	15.43	8.67
RED	**10.04**	**28.29**	**24.65**	**11.99**	**27.97**	**28.53**
Transformer	2.33	19.88	13.27	11.45	25.19	20.92
Outstanding Balance						
Baseline	5.18	24.78	9.23	1.87	13.68	8.52
RED	23.10	43.28	**47.89**	**12.51**	28.47	**31.35**
Transformer	**30.35**	**44.65**	40.84	12.07	**28.75**	25.02
Validation of DL						
Baseline	1.28	18.11	10.74	2.47	15.03	8.52
RED	**9.35**	**28.12**	**23.79**	14.93	**31.82**	**34.25**
Transformer	2.12	19.25	12.45	**15.08**	28.33	24.74
Account Deactivation - EN Validation of address - DE						
Baseline	5.74	22.75	9.07	4.13	18.11	8.93
RED	**29.43**	**46.64**	**41.04**	**21.33**	**44.30**	**45.69**
Transformer	1.28	21.51	14.02	14.65	37.72	31.60

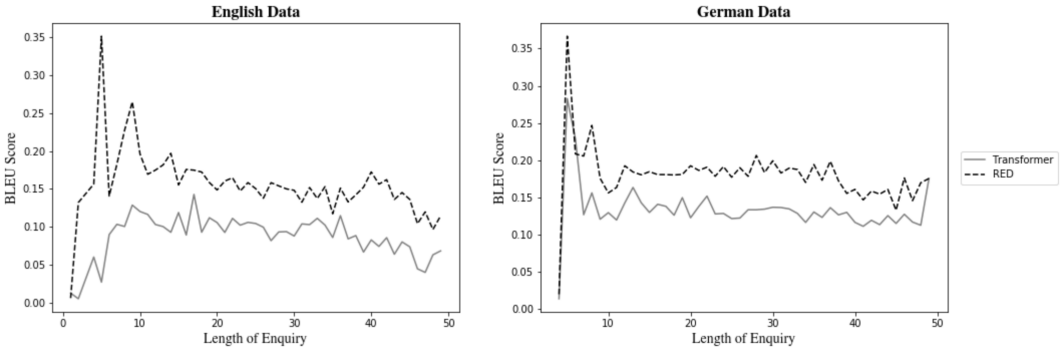
DL: Driver's Licence
BLEU: $= min(1 - \frac{r}{c}, 0) + \sum_{n=1}^{N} \frac{1}{N} log(p_n)$
ROUGE-1:$= min(1 - \frac{r}{c}, 0) + \ log(re_1)$
METEOR: $= min(1 - \frac{r}{c}, 0) + \ \sum_{n=1}^{N} \frac{1}{N} log(\frac{10 \cdot re_n \cdot p_n}{r_n + 9p_n})$
where r: reference length, c: candidate length, p_n: n-gram precision, re_n: n-gram recall
All evaluation metrics are referenced with an actual human response.

To summarize, while we do see wild swings in performance for the English data set the metrics indicate a more similar performance across topics in the German data set. Consequently, it is sensible to look at the decoding behavior considering certain characteristics of the data. One candidate driver of the quality of automated responses could be the length of the customer enquiry. As is evident from Figure 3 topics vary by the average message length. As previously mentioned, each neural model decodes a target output conditional on a given input. It seems likely that the neural models might struggle to produce topical responses to longer enquiries which contain a more complex subject matter. On the other hand, they might perform relatively better on shorter enquiries with a simple subject matter. Similarly, model performance will likely variate with the length of the reference response on which the models are evaluated. Intuitively, as the length of the reference response increases, it would seem more unlikely that the neural models will be able to produce a similar output as the length of dependencies increases. As we can see from Figure 4, the results vary for the two data sets. First, in the English data set the quality of responses seems to decrease as the enquiries get longer. Conversely, in the German data set, other than for very short enquiries, the quality of responses does not fluctuate with the length of the input sentence. Furthermore, with respect to the length of the reference responses it is easy to see that the performance of Transformer model peaks for sentences of lengths 17 and 37 but otherwise remains rather flat. In contrast to expectations, the performance in RED model seems to improve as the length of the references increases. This confirms the results of the Transformer model in our topic analysis, as the Transformer only seems capable of answering a certain type of question but otherwise is unable to replicate human response behaviour. Moving over to the German data set the behavior of the two frameworks is somewhat more similar. For example, both models peak at a sentence length of 24 after which performance seems to decrease again more inline with previous expectations. Regardless, as mentioned by Liu et al. (2016) the behavior of natural language generation system

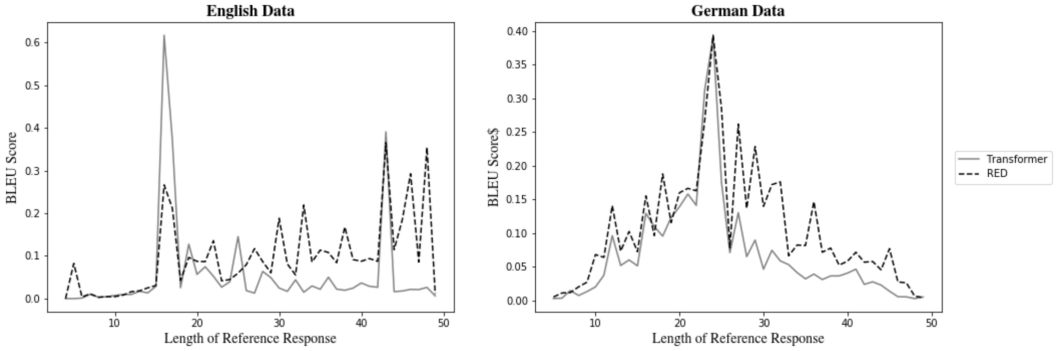
Figure 4

BLEU Scores by Sequence Length

(a) *Length of Enquiries*



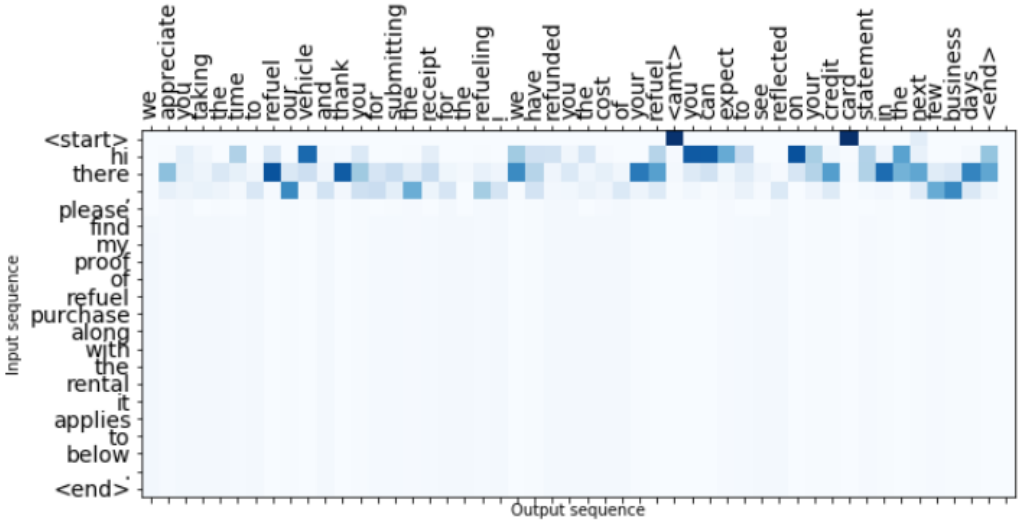
(b) *Length of References*



should not be solely explained through the use of automatic evaluation metrics. Moreover, it has been often argued that further development and understanding of machine learning, and especially deep learning models, requires certain level of interpretability (Doshi-Velez and Kim, 2017). As discussed by Xie (2017), a common tool for understanding the behavior of attention-based NLG frameworks is to look at the attention scores during decoding. More specifically, this should give us a better understanding of how the decoder links the generated output to the input sequence. Figure 5 visualizes the attention score vectors which are fed to the decoder at each step. We use a heat map to identify higher attention scores which are shaded in a deeper blue. In addition, the y-axis displays the input sequence that prompts the response displayed on the x-axis. Immediately, we see that the attention mechanism only captures the relationship between key words in the response with the beginning of the input sequence. It is possible

Figure 5

Recurrent Encoder-Decoder: Dot Attention

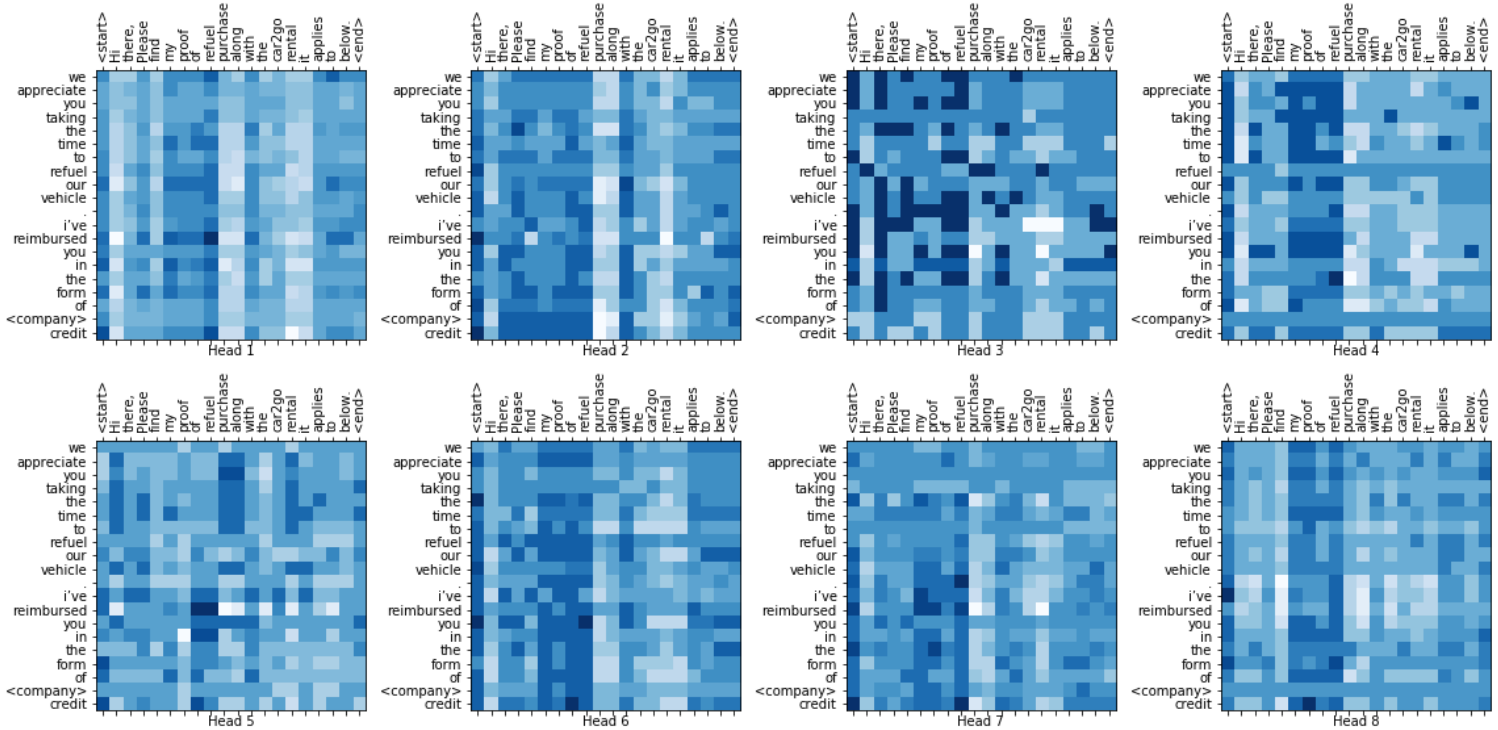


that the attention mechanism is too simplistic to capture the complex semantic and logical relationship between input and response. In the beginning, RED frameworks were implemented for translation tasks, e.g. translating sequences from one language into another. Typically, in machine translation applications this type of attention attention mechanism is very successful in capturing the dependencies between input and output. A likely explanation is that in translation applications the alignment of words similar in both input and output. For example, the fifth word in the first sentence is a direct translation of the fifth word in the output sentence. As a consequence, alignment in translation tasks is much simpler than in NLG applications where the distance between related parts of the input and output sequences can be rather large. On a more positive note, the mechanism clearly identifies the key words of the decoded sentence e.g. the money token $< amt >$, refuel and vehicle. Evidently, successful attention methods in RED frameworks from the MT literature cannot the applied as is to dialogue generation. Even tough our automatic evaluation metrics indicate that the our Transformer model is inferior, we investigate its decoding behavior in a similar manner. Admittedly, this is not as straight forward as in our RED frame-

work, as each of our Transformer model's six decoder layers has eight separate attention heads. Conversely, one would assume that the model's ability to attend to different positions in the input sequence in parallel should enable it to capture the semantic and logical dependencies between input and output. Thus Figure 6 depicts the attention output of all eight attention heads from our model's fifth decoder layer for the same example input as in Figure 5. The input sequence is depicted on the x-axis and the produced output sits on y-axis. At first glance it is obvious that unlike in the RED model the Transformer model is able to relate each position in the input sequence to a step in the decoded output sentence. In addition, the content of the output response is grammatically sound and on topic. However, there is no discernible pattern to the relationship the model establishes between input and output sentences. It remains an open question if this is due to a misspecification of the model, underfitting on the training data or if the Transformer is simply unsuitable for this NLG application. For instance, in the use of a Transformer model of Rashkin et al. (2018) the embedding layer consisted of a dialogue state embedding in addition to the word and positional embeddings. Their intent was to capture the personal attributes within a multi-turn chat dialogue. Also, they used a pre-trained model which was fine-tuned on their data set. In contrast, the model in this work was only trained on the closed domain data set since any pre-training would have made it more difficult for the model to return the customer service responses word for word. Overall, the picture that ermeges from the results is twofold. For one, both neural models clearly outperform the benchmark tf-idf IR set up. This shows that both are systems are useful for creating customer service email responses for which the common critique of bland and dull outputs is not a critical issue. Nonetheless, the decoding behavior shows that these frameworks may excel at MT but need fine-tuning if they are to implemented for dialogue applications in production systems without oversight. In summary, these frameworks may be suited more to suggestion not as independent response agents.

Figure 6

Transformer: Multi-Head Attention



6 Human Evaluation

Addressing the criticisms of Liu et al. (2016) on the use of automated evaluation metrics this work follows Adiwardana et al. (2020) in adopting a small scale human evaluation of sample outputs in order to measure the quality of the model output. Furthermore, it is possible to compare the results of the manual evaluation with the results from the automated measures in order to validate their use. The evaluation proceeded as follows: 20 customer service professionals (i.e. agents) were given 20 outputs from the RED model on example inputs randomly drawn from the English test set. Hence, the use of agents adds validity to the evaluation due to their familiarity with the domain. Only examples from the English data set were used. For efficiency proposes, most agents are trained on multiple languages but all are trained in English. The agents were asked to evaluate the model output for two criteria. First, they were asked whether the produced output was sensible in a manner that the output produced was grammatically correct, contained no repetitions and was logically sound. Secondly, agents were asked to ascertain if the output matched the topic of the customer enquiry or was specific with regards to the subject matter. Naturally, results from such a relatively small sample size are subject to large standard deviations and as such one should be careful in extrapolating results onto the entire test set. Regardless, the results for the agreement rate shown in Table 5 for both the sensibleness as well as the specificity criteria are quite high. For example, in 57% of evaluations by customer service experts, the agents agreed that the output generated by the RED model was sensible and in more than 60% of evaluations the output was rated as specific to the customer enquiry. In the end, the response suggestion from the RED model were rated both specific and sensible in 44% of evaluations. Moreover, the sensibleness and BLEU criteria are significantly and positively correlated providing some support for the argument that the use of automated metrics of linguistic quality in a customer service application is valid. However, our specificity criteria and the BLEU scores show no positive correla-

Table 5
Results: Expert Evaluation

	Human Evaluation		Spearman Rank Correlations		
Criteria	Agreement (%)	Standard Deviation	Sensibleness	Specificity	BLEU
Sensibleness	57.00	± 23.02	-	0.5454**	0.4267*
Specificity	60.75	± 30.10	0.5454**	-	0.071
Combination	44.00	± 24.31	0.8874***	0.8195***	0.3141

Significance Levels. * $p < .1$, ** $p < .05$, ***$p < .01$ 20 sample responses evaluated by 20 agents
Question 1: Is the generated output sensible i.e. grammatically correct, not repetitive and logically sound ?
Question 2: Is the generate output specific i.e. does it address the customer's issue ?

tion at all. Consequently, it would have been interesting to conduct a follow up questionnaire asking agents which topic they believed the model answered and to which they would assign the customer enquiry. Thus it would allow the construction of a map of classification errors as it is possible that the model may be to generic in its output (Li et al., 2015). Nonetheless, the results also show that the two manual evaluation criteria are strongly correlated but not perfectly so which indicates our survey indeed measures two related but separate aspects of output quality. Overall, the results from the manual evaluation support the results from the automated evaluation metrics. Specifically, a neural response model is capable of generating quality responses in customer service even though it cannot be guaranteed that the output always meets a high quality standard.

7 Conclusion

To summarize, this thesis has shown that neural response models are capable of outperforming the industry standard information retrieval frameworks which select response candidates from pre-determined sets when generating responses in a customer service application. Additionally, using measures based on word overlap a Recurrent Encoder-Decoder Network outperformed a Transformer model where the performance of both frameworks varied with the subject matter of the customer enquiry. Furthermore, a small sample evaluation conducted with customer service experts showed a positive link between our word overlap mea-

sures and human judgments of the linguistic quality of the responses generated by a Recurrent Encoder-Decoder Network. However, it remains an open question why word overlap measures indicate variation across topics and how come human judgements on specificity appear unrelated to word overlap metrics. The variation of performance measures across topics and the unsatisfying decoding behavior indicated by the attention plots, point to an open question which the Literature Review Table 1 already hints at. Namely, whether in order for neural response frameworks to reliably generate specific and sensible responses, it is necessary increase the number of model parameters, the size of training data sets or add structural components such as memory or a different loss function. On the one hand, Adiwardana et al. (2020) have shown that frameworks of the class of Evolved Transformers introduced by So et al. (2019) can push the performance boundaries of open-domain conversational models. For Evolved Transformers the architecture is trained by neural network instead of being defined by a developer or being determined in a randomized parameter search. On the other hand, the relatively worse performance of our Transformer model and the bad behavior of its attention mechanism may point to the need for additional structural components in order for neural response models to be able to reliably recognize the intent captured in the input sequence and to relate it to the generated output during decoding. To answer this question it will be interesting to evaluate the performance of neural response models such as Adiwardana et al. (2020) in future research on smaller closed-domain data sets like the one used in this thesis. From a practitioner's perspective these frameworks do not seem ready for productive use as independent answering agents in a customer service environment. Currently, these frameworks are more likely to be useful as auto compose tools for customer service agents helping them to finish responses to questions more quickly and efficiently. In comparison to the prevalent use of information retrieval mechanisms to select pre-written templates, this offers customer service agents more flexibility and would likely be more accepted by workers in the industry.

References

Adiwardana, D., Luong, M.-T., So, D. R., Hall, J., Fiedel, N., Thoppilan, R., Yang, Z., Kulshreshtha, A., Nemade, G., Lu, Y., and Le, Q. V. (2020). Towards a human-like open-domain chatbot.

Aksin, O., Armony, M., and Mehrotra, V. (2007). The modern call center: A multi-disciplinary perspective on operations management research. *Production and Operations Management*, 16:665 – 688.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Bain, P., Watson, A., Mulvey, G., Taylor, P., and Gall, G. (2002). Taylorism, targets and the pursuit of quantity and quality by call centre management. *New Technology, Work and Employment*, 17(3):170–185.

Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and

Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Dale, R. (2016). The return of the chatbots. *Natural Language Engineering*, 22(5):811–817.

Doshi-Velez, F. and Kim, B. (2017). Towards a rigorous science of interpretable machine learning.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179 – 211.

Gao, J., Galley, M., and Li, L. (2018). Neural approaches to conversational AI. *CoRR*, abs/1809.08267.

Henderson, M., Al-Rfou, R., Strope, B., Sung, Y., Lukács, L., Guo, R., Kumar, S., Miklos, B., and Kurzweil, R. (2017). Efficient natural language response suggestion for smart reply. *CoRR*, abs/1705.00652.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Holman, D., Chissick, C., and Totterdell, P. (2002). The effects of performance monitoring on emotional labor and well-being in call centers. *Motivation and Emotion*, 26:57–81.

Huang, L., Zhao, K., and Ma, M. (2018). When to finish? optimal beam search for neural text generation (modulo beam size). *CoRR*, abs/1809.00069.

Hui, S. and Jha, G. (2000). Data mining for customer service support. *Information and Management*, 38(1):1 – 13.

Ji, Z., Lu, Z., and Li, H. (2014). An information retrieval approach to short text conversation. *CoRR*, abs/1408.6988.

Józefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Kalchbrenner, N., Espeholt, L., Simonyan, K., van den Oord, A., Graves, A., and Kavukcuoglu, K. (2016). Neural machine translation in linear time. *CoRR*, abs/1610.10099.

Kannan, A., Kurach, K., Ravi, S., Kaufmann, T., Tomkins, A., Miklos, B., Corrado, G., Lukács, L., Ganea, M., Young, P., and Ramavajjala, V. (2016). Smart reply: Automated response suggestion for email. *CoRR*, abs/1606.04870.

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2015). A diversity-promoting objective function for neural conversation models. *CoRR*, abs/1510.03055.

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2016). A persona-based neural conversation model. *CoRR*, abs/1603.06155.

Li, J., Monroe, W., Shi, T., Ritter, A., and Jurafsky, D. (2017). Adversarial learning for neural dialogue generation. *CoRR*, abs/1701.06547.

Lin, C.-Y. (2004). ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Liu, C.-W., Lowe, R., Serban, I., Noseworthy, M., Charlin, L., and Pineau, J. (2016). 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*, pages 2122–2132, Austin, Texas. Association for Computational Linguistics.

Luan, Y., Ji, Y., and Ostendorf, M. (2016). LSTM based conversation models. *CoRR*, abs/1603.09457.

Luong, M., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *INTERSPEECH*.

Novikova, J., Dušek, O., Cercas Curry, A., and Rieser, V. (2017). Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). 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. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Rashkin, H., Smith, E. M., Li, M., and Boureau, Y. (2018). I know the feeling: Learning to converse with empathy. *CoRR*, abs/1811.00207.

Reiter, E. (2018). A structured review of the validity of BLEU. *Computational Linguistics*, 44(3):393–401.

Robertson, S. (2004). Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation - J DOC*, 60:503–520.

Salesforce Research (2019). 3rd state of service report.

Santhanam, S. and Shaikh, S. (2019). A survey of natural language generation techniques with a focus on dialogue systems - past, present and future directions. *CoRR*, abs/1906.00500.

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2015). Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808.

Serban, I. V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A. C., and Bengio, Y. (2016). A hierarchical latent variable encoder-decoder model for generating dialogues. *CoRR*, abs/1605.06069.

So, D. R., Liang, C., and Le, Q. V. (2019). The evolved transformer. *CoRR*, abs/1901.11117.

Sparck Jones, K. (1988). *A Statistical Interpretation of Term Specificity and Its Application in Retrieval*, page 132–142. Taylor Graham Publishing, GBR.

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). Weakly supervised memory networks. *CoRR*, abs/1503.08895.

Sutskever, I., Martens, J., and Hinton, G. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 1017–1024, USA. Omnipress.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

van der Maaten, L. and Hinton, G. (2008). Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Vinyals, O. and Le, Q. V. (2015). A neural conversational model. *CoRR*, abs/1506.05869.

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

Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. *CoRR*, abs/1410.3916.

Williams, J. D., Kamal, E., Ashour, M., Amr, H., Miller, J., and Zweig, G. (2015). Fast and easy language understanding for dialog systems with Microsoft language understanding intelligent service (LUIS). In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 159–161, Prague, Czech Republic. Association for Computational Linguistics.

Wolf, T., Sanh, V., Chaumond, J., and Delangue, C. (2019). Transfertransfo: A transfer learning approach for neural network based conversational agents. *CoRR*, abs/1901.08149.

Xie, Z. (2017). Neural text generation: A practical guide. *arXiv preprint arXiv:1711.09534*.

Declaration of Authorship

I hereby confirm that I have authored this thesis independently, other than by the use of the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such. Furthermore, I am aware of the University's regulations concerning plagiarism, including those regulations concerning disciplinary actions that may result from plagiarism.

Berlin, February 25$^{\text{th}}$, 2020 Sydney Richards