BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

STYLIZED NATURAL LANGUAGE GENERATION IN DIALOGUE SYSTEMS

GENEROVÁNÍ STYLIZOVANÉHO LIDSKÉHO JAZYKA V DIALOGOVÝCH SYSTÉMECH

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **KSENIA BOLSHAKOVA**
AUTOR PRÁCE

SUPERVISOR **Ing. MARTIN FAJČÍK**
VEDOUCÍ PRÁCE

BRNO 2020

Abstract

In this thesis, the stylistic generation of natural language in dialogue systems is explored. The goal is to provide insight into the current state of research in this area and find promising research directions to the future. Feature-based modifications and a model for weighted decoding were implemented. The realization of the model is presented in two versions. The first one is a LSTM-based baseline and the second one uses state-of-the-art pretrained models for text generation. The experiments showed that when combining models, trained on certain styles, by weighted decoding, it is possible to achieve the generation of stylistic text, but there is a problem with the context. Based on these findings, recommendations and possible future research areas are suggested.

Abstrakt

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém (slovenském) jazyce.

Keywords

neural network, natural language generation, dialogue systems, seq2seq, BART, GPT-2, LSTM

Klíčová slova

Sem budou zapsána jednotlivá klíčová slova v českém (slovenském) jazyce, oddělená čárkami.

Reference

BOLSHAKOVA, Ksenia. *Stylized Natural Language Generation in Dialogue Systems.* Brno, 2020. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Martin Fajčík

Stylized Natural Language Generation in Dialogue Systems

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Martin Fajčík. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Ksenia Bolshakova
May 9, 2020

Acknowledgements

I would like to thank my supervisor Ing. Martin Fajčík for his guidance, constructive feedback and help with the thesis.

Contents

Chapter 1

Introduction

Dialogue system (DS) is a computer system which interacts with a human in natural language. These systems are used in cars (hands-free car-specific functions, Android Auto, Apple CarPlay), web, robots, computer games etc, because a conversation is a natural way for people to get information.

Natural Language Generation (NLG) is an important component of dialogue systems. NLG goal is to imitate human behaviour, what is very important for dialogue systems. DS can be classified into task-oriented, which focused on completing a certain tasks and adhere to a determined script for each stage of the conversation, and non-task-oriented, which do not have a stated goal to work towards. A lot of devices today have incorporated intelligent agents, such as Yandex's Alisa, Apple's Siri, Microsoft's Cortana, Amazon Alexa, and Google Assistant. These are in fact dialogue systems. Task-oriented dialogue acts make conversations more interpetable and controllable. On the other hand, they also hinder scaling such systems to new domains (i.e. conversation topics). To escape from the limitation, recent interest of research started moving to non-task-oriented chitchat dialogues (chatbots). Chitchat dialogues are systems designed to mimic the unstructured human-human conversation. This kind of dialogue systems often have an entertainment value, such as Cleverbot, Microsoft's XiaoIce system etc. Chatbots have also been used for testing theories of psychological counseling [49].

The ability to communicate freely in a natural language is one of the hallmarks of human intelligence, and is likely one of the requirements for true artificial intelligence. Many researchers work on an open-ended (i.e. there is a huge range of appropriate outputs given the input) chitchat dialogues to explore this aspect of intelligence. In task-oriented dialogue systems there is a relatively narrow range of correct outputs given the input. An example of possible responses to an input sequence is shown in the Figure 1.1, but there are still many ways to answer the question "Hi, how are you?". All answers have different attributes of generated text, such as sentiment, lengths of answers and specificity, which we as humans can control.

Creating a non-task-oriented DS is a challenge for researchers, because there are a lot of topics of conversations as well as user reactions and responses to them. Such bots often suffer from the inability to generate human-like conversations. Their replies are often too generic, because non-specific responses are in natural language (e.g. "Ok, I see", "I don't know").

According to [40] one of the most important cognitive behaviors in humans is expressing and understanding emotions. That is why it is necessary to pay attention not only to generation of a semantically and syntactically correct text, but also to the emotions and

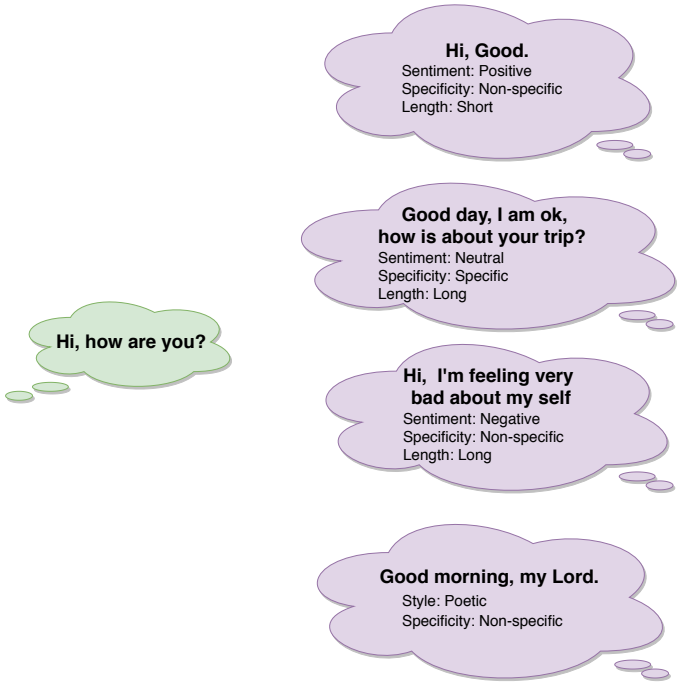

Figure 1.1: Example of answers to the input sequence with setted attributes.

language style in which person communicates to make a dialogue more diverse and interesting. Carefully formulated speech without cliches or jargon is essential to avoid inaccurate presentation and ensure effective communication. Most of the modern generative models are trained on huge corpora which include different contributions from various authors. Texts produced with such models are often not perceived as natural and characterized as non-human, because humans have recognizable writing and communication styles.

The main purpose of this thesis was to create a dialogue system, which is able to generate text in different styles and control the manifestation of each style. Chapter 2 presents dialogue systems and NLG problems that arise during a dialogue. Chapter 3 contains the overview of concepts used in the Natural Language Generation. It starts off with an explanation of fundamental techniques common to a lot of machine learning approaches and then continues to describe models specific for NLG and used in the experiments in this thesis. Chapter 4 contains a portfolio of current methods and models for solving the problems described in the chapter 2. Chapter 5 describes evaluation methods for NLG in dialogue systems and publicly available datasets, which are used for the models training in the thesis. Chapter 6 presents designed solution for stylized natural language generation and its implementation. Chapter 7 contains the results of the experiments, their analysis and evaluation of the designed model. Chapter 8 summarizes the results reported in the previous chapter and providing recommendations on future research in stylized natural language generation in dialogue systems, which are based on the thesis' results.

Chapter 2

Dialogue systems

NLG is an important component of a DS. According to [1] Natural Language Generation is defined as "the process by which thought is rendered into language". NLG approaches can be grouped into two categories, one focuses on generating text using templates or (linguistic) rules (i.e. data-to-text generation), the other uses corpus-based statistical methods (i.e. text-to-text generation), where corpus is a collection of texts [31]. Spectrum of these approaches along with its (dis)advantages is represented in the Figure 2.1.

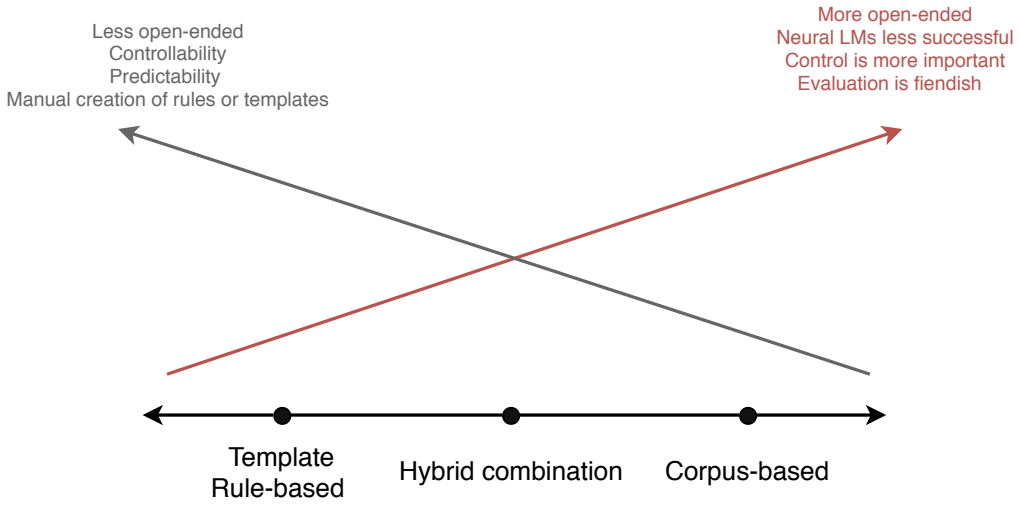

Figure 2.1: Spectrum of the NLG approaches.

2.1 Template-based approach

Until recently Natural Language Generation component of a dialog system used primarily hand-coded generation templates, which represented model sentences in a natural language mapped to a particular semantic content. The template-based system selects a proper response for the current conversation from a repository with response selection algorithms. Templates are often designed for a specific task in a given domain [28]. Example of template-based system is shown in the Table 2.1.

Example:
User's input: "I'm going to travel from Moscow on April 2."
Template: What time would you like to travel from {*departure_city*} on {*departure_date*}? Repository: *departure_city*: {Moscow, Brno, Prague}, *departure_date*: {April 2, March 3}
Agent's output: "What time would you like to travel from Moscow on April 2?"

Table 2.1: The example of template-based approach.

Advantages of template-based approach

The pros of template-based approach include that the output produced by this approach is likely to be grammatically correct and not contain unexpected generation errors. The another one is that the process of sentence generation is fully controlled, these models are robust and reliable because they consist of clearly defined rules.

Disadvantages of template-based approach

These models require time and human resources to deploy a real dialogue system, because templates are constructed manually, and the number of templates grows quickly (using different templates for singular and plural versions). The next disadvantage is that human created templates often sound unnatural due to their generic structures. Another weak points are that template-based systems are not able to handle unknown inputs and cannot make variation in output, it is just concatenation of strings. This approach also is not flexible, because it has limits to use templates in other domains. The cons of template-based approach is that a template-based model is not able to learn and is not able to adapt to the user, that's why it generates rigid and stylised responses without the natural variation of human language.

2.2 Corpus-based approach

Corpus-based systems dominate in the NLG community, especially in the case of open-ended tasks, where it is almost impossible to hand-craft the templates for all possible combinations of semantic units. Corpus-based systems include statistical and machine learning approaches to resolve problems of the template-based approach [38]. Corpus-based approach mines large datasets of human-human conversations.

Advantages of corpus-based approach

Corpus-based models have ability to generate more proper responses that could have never appeared in the corpus. The next advantage is the ability to mimick the language of a real domain expert and use these models for open-domain dialogue systems. The benefits of corpus-based approach also include that dynamic approach is able to learn and to handle unknown inputs, it is also has a lot of possible variations of output.

Disadvantages of corpus-based approach

It is necessary to have a corpus, which contains a large amount of data on a variety of topics to get a sensible output. Even if the corpus is available, process of text generation is not fully controlled and the output can be incorrect or does not need to make a sense. This approach still has a lot of problems, what will be described in more detail in the section 2.4.

2.3 Language Models

Corpus-based systems usually rely on **Language Models(LMs)** to generate sequences of texts. LM is a probabilistic model which estimates the probability of a sequence of words. The Equation 2.1 represents the language model, where W is a sequence and $w_1, w_2, ..., w_n$ are words in this sequence.

$$P(W) = P(w_1, w_2, ..., w_n) \tag{2.1}$$

The **Chain rule** (Equation 2.2) is commonly employed to factorize the joint probability of a sentence into the product of the conditional probabilities (Equation 2.3) of a word given previous words.

$$P(w_1, w_2, ..., w_n) = \prod_i P(w_i | w_1, w_2, ..., w_{i-1}) \tag{2.2}$$

$$P(A|B) = P(A \cap B)/P(B) \tag{2.3}$$

In Equation 2.3 $P(A \cap B)$ is the probability that both events A and B occur.

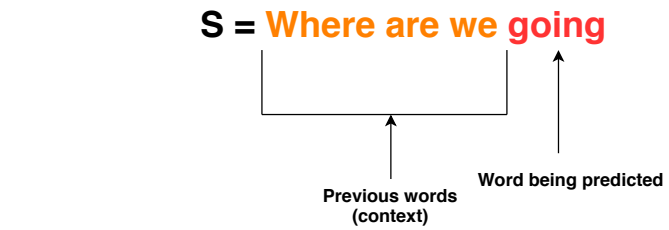

Figure 2.2: Example of calculating a sentence probability by using chain rule.

An example in the Figure 2.2 shows how to predict probability of a word given previous words. A subsequence (context) may consist of a very large number of words and the likelihood that such subsequence is found in a corpus is very small. It is a main problem in language models, which is called **data sparsity**.

Data sparsity is the phenomenon of not observing enough data in a corpus to model language accurately. The solution to resolve this issue is to make the assumption that the probability of a word depends only on the previous n words and use **N-gram model** (N-gram is a sequence of N words).

The n-gram "She is studying IT" from the Table 2.2 does not occur as often in texts of corpus as n-grams "Hi", "New York" and "The Three Musketeers". Knowing a probability to the occurrence of an N-gram in a sequence of words can be useful, because it can help

Hi	1-gram
New York	2-gram
The Three Musketeers	3-gram
She is studying IT	4-gram

Table 2.2: The example of N-grams.

to decide which N-grams can be chunked together to from single entities (like "New York" chuncked together as one word). It can also help make next word predictions. For example, "tea" is more likely than "ball" in the phrase "I would like to drink".

According to [4] the state-of-the-art way to fight with data sparsity is learning a distributed representation for words, which allows each training sentence to inform the model about an exponential number of semantically neighboring sentences. The model learns simultaneously a distributed representation for each word along with the probability function for word sequences. A sequence of words that has never been seen before gets high probability if words in this sequence are similar in the sense of having a nearby representation to words forming an already seen sentence. Authors of [4] used neural networks (artificial neural networks are described in the section 3.1) for the probability function. The proposed approach improved n-gram models and took an advantage of longer contexts.

2.4 Flaws of dialogue systems

The ability to communicate with machines in a natural language is a long-standing dream of a mankind. Today's dialogue systems often encounter criticism. There are many scientific works on creating more natural dialogue systems. Markus M. Berg defines a natural dialogue system in [6] as "a form of dialogue system that tries to improve usability and user satisfaction by imitating human behaviour". It affects the features of human-to-human dialogue (for example, topic changes, sub-dialogues) and seeks to integrate them into dialogue systems for human interaction with the machine. Open-ended natural dialogue systems still have flaws in generating a response to the user.

#	**Example**	**Problem type**
1	While Bob ate an apple when it swims.	adequacy
2	Why a mouse when it spins?	adequacy
3	-Yes, I'm studying law at the moment. -Good. -I like playing the piano. -Good.	repetition
4	-Do you go get coffee often? -I am a musician.	response-relatedness
5	-What is your favorite film? -I do not know -What is your hobby? -I have no idea	specificity

Table 2.3: Examples of DS problems.

Main problems of dialogue systems are represented in the Table 2.3. A problem of adequacy shows that a response can be grammatically and syntactically composed correctly, but this sentence does not make sense. A problem of repetition makes conversation boring. A problem of response-relatedness shows that the answer to the question does not make a sense in this context and it spoils the impression of the conversation. A general and trivial response ("I don't know") in a dialogue could correspond to a very large variety of input sequences, but this does not make the conversation informative and specific.

As noticed in [46], the main task of NLG is to select, inflect and order words "to communicate the input meaning" as completely, clearly and fluently as possible in context. That's why it is necessary to control if output is appropriate or felicitous in a given context. A good generator usually relies on several factors:

- **adequacy** (a sentence that is ambiguous or not contains communicates meaning in the input, is **not** adequate)
- **repetition** (self-repetition across sequences and with sequences, repeating the conversational partner)
- **response-relatedness** (efficacy in context)
- **specificity** (informativeness in a dialogue)
- **variation** (there are 2 basic forms of variation: *word choice variation* and *word order variation* for enriching speech)

#	**Example**
1	I bought movie tickets on Tuesday.
2	I got movie tickets on Tuesday.
3	On Tuesday I bought movie tickets.
4	On movie Tuesday tickets I bought.
5	I bought tickets for the Tuesday movie.

Table 2.4: The example of sentences' variation.

An example in the Table 2.4 shows all types of variation. Sometimes this factor can be syntactically incorrect or unclear, what you can see in the forth sentence. In fifth sentence a variation changed the meaning of part of the sentence. In addition, the variation may add or remove meaning possibilities.

One of the hardest problem in text generation is a language style, which makes a response to an user more human. This task is challenging due to the difficulty of capturing emotional factors and the complex mechanism of human emotions. Some people use obscene speech, some use a lot of expressive means, jargon or jokes to make speech more emotional. This is what distinguishes people and makes their communication more interesting.

Chapter 3

Concepts in Natural Language Generation

Development of NLG from static template-like generation to dynamic generation of sentences took a lot of time and models developed along with it. Corpus-based generation uses a generative probabilistic model what can be implemented in many ways. The model focuses on response generation in the context of dialogue, where the task is to generate a response, given an input sequence. Thus, these models fit well within the sequence-to-sequence (seq2seq) (i.e. encoder-decoder) models, which are described in more detail in section 3.4. Neural networks are explained for a better understanding seq2seq model.

3.1 Neural Networks

Artificial Neural Networks are inspired by biological neural networks. Artificial neuron (Figure 3.1) is a computational unit in an artificial neural network with a set of real-valued inputs $x_1, x_2 ... x_n$ and an output y, where each input x_i has a corresponding weight w_i. Weights determine the influence of the input on the output. The neuron's output is the weighted sum of its inputs, which are passed through a non-linear function known as an activation function or transfer function. The transfer functions usually have a sigmoid shape and model the threshold for neuron firing. Bias is an additional parameter in the neural network, which is used to adjust the output along with the weighted sum of the inputs to the neuron. Bias value allows to shift the activation function.

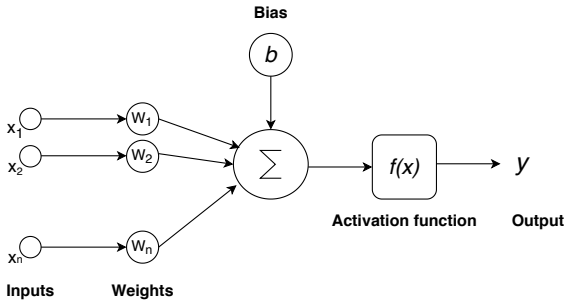

Figure 3.1: Architecture of an artificial neuron.

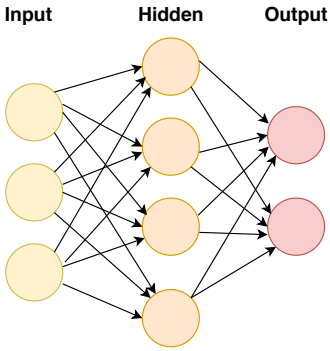

Figure 3.2: The fully-connected neural network with 1 hidden layer.

Neurons' outputs can be connected to inputs of other neurons and the calculation is propagated through the network. In fully connected NNs neurons are organized into layers where generally the output of the layer is the input for a next layer. The Figure 3.2 represents the most common type of layer – the fully-connected layer where all neurons between adjacent layers are connected with each other.

The goal of neural network training is to minimize an error (i.e. loss) between predicted outputs of NN and the real data with help of **Gradient Descent** algorithm [22]. Gradient Descent finds a local minimum of a differentiable function. **Backpropagation** [39] is used for computing a gradient descent with respect to the weights of NN and for fine-tuning weights. Tuning of the weights ensures lower error rates, what makes a model to approach the true distribution of the data.

3.2 Elman Networks

Elman Networks are a simple form of recurrent neural networks (RNN) used in natural language processing (NLP) as they allow modelling temporal depedencies of variable length in the data (like context) to be captured. RNNs share the same structure as NNs described in the section 3.1, except each layer also has an internal state (i.e. hidden state), which captures information about the previous layer inputs. This allows the network to keep track of past data while processing current inputs.

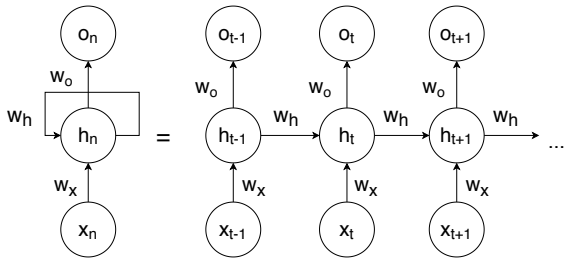

Figure 3.3: Architecture of Elman neural network, where variable **h** is a hidden state, **x** is an input and **o** is an output. **w** is a weight, which is optimised to produce a sensible output.

The architecture of Elman Networks is illustrated in the Figure 3.3. A hidden state **h** is a vector, which calculated from the input x and the previous hidden state. An output **o**

is calculated from this new hidden state. The Equations 3.1 and 3.2 show the formulas for a traditional recurrent neural network.

$$h_t = \sigma(W_h h_{t-1} + W_x x_t) \tag{3.1}$$

$$o_t = softmax(W_o h_t) \tag{3.2}$$

The RNN-based models have been used for NLG as a component of end-to-end trainable task-oriented dialogue system [51].

Nowadays traditional RNN networks almost are not used in NLG, because they have problems with vanishing and exploding gradients. As introduced in [5] these problems occur due to instability of the spectral norm of the Jacobian matrix during training. It happens, because long term components can grow exponentially more then short term ones. The vanishing gradients problem refers to the opposite behaviour. The long term components go exponentially fast to spectral norm 0, which makes it impossible for the model to learn correlation between temporally distant events. This issue has motivated researchers in development of more advanced RNNs like the LSTM [15].

3.3 Long short-term memory (LSTM)

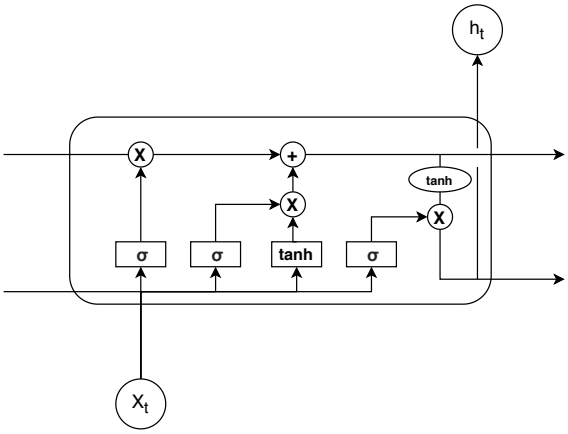

Figure 3.4: A cell in an LSTM network.

LSTM networks are a special kind of RNN, which reduce the vanishing gradient problem. It makes them much more effective on capturing long-term dependencies. All recurrent neural networks have the form of a chain of repeating modules of neural network. LSTM network also has this chain structure, but the repeating modules (cells) has a different structure than simple RNN. The key of the solution is usage of multiple gates and a cell state, which runs through all the cells and is manipulated using these gates – parts of the state may be added or removed. Each gate is a sigmoid layer that outputs a number between 0 and 1, which represents the degree of the cell state modification.

The Figure 3.4 represents a structure of a LSTM cell. LSTM cell processes data sequentially for 1 time step and keeps its hidden state through time. First, the network decides how much of information from previous steps to keep stored in its cell state, by using the forget gate, which consists of a sigmoid function applied to weighted sum of previous output, input and bias (Equation 3.3). W are updated through the backpropagation algorithm

weights, b_f is a bias, x_t is an input, h_{t-1} is a hidden state from previous step.

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{3.3}$$

The next step is to decide how much of the inner state is going to be updated (i.e. what part of the result the cell is going to store in its state), by using input gate (Equation 3.4).

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{3.4}$$

After calculating the state modification, it is necessary to compute the new values (i.e. candidate values) which will be stored in it, by using activation function (Equation 3.5).

$$\tilde{c_t} = tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{3.5}$$

Updating the cell state is based on the previous state and the candidate values (Equation 3.6). $\odot$ denotes the Hadamard product (element-wise product).

$$c_t = \tilde{c_t} \odot i_t + c_{t-1} \odot f_t \tag{3.6}$$

Output gate is represented in Equation 3.7.

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{3.7}$$

And the final step producing the hidden state for the next timestep. It is based on the newly updated cell state, transformed by *tanh* function and multiplied by the output gate (Equation 3.8).

$$h_t = o_t \odot tanh(c_t) \tag{3.8}$$

This model does not suffer so much from a vanishing gradient as simple RNN networks, but still the capacity of the LSTM memory is limited and high computational requirements make learning LSTM difficult.

3.4 Sequence-to-sequence model (seq2seq)

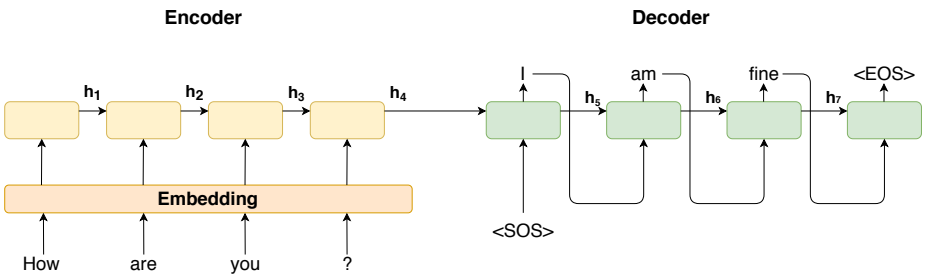

Figure 3.5: The architecture of sequence-to-sequence model. h_i represents a hidden state.

Seq2seq models were introduced in 2014 [47]. This model uses an encoder-decoder architecture (Figure 3.5). Both the encoder and the decoder are recurrent neural networks (vanilla version of RNN is rarely used, because of the problems described in the section 3.2). The role of the encoder is to encode the input, a sequence of variable length data, to a fixed length vector. Decoder based on this vector generates an output sequence of data of different length. These two neural networks are connected into one model to maximise

the learning effect. Seq2seq model is very effective in solving NLP problems with long-term dependencies, because input and output sequences can have different lengths and recurrent neural networks can work with "arbitrarily long" context. This kind of problems do often appears in machine translation, text summarization, dialogue systems etc.

The Figure 3.5 presents traditional encoder-decoder arhitecture. Encoder converts an input sequence of words to a corresponding fixed-size hidden vector. Each vector represents the current token and the context of it.

Every time step, it takes a vector that represents a word and pass its output to the next layer. The last hidden state of encoder passes its output to the first layer of the decoder. The final hidden state of the encoder is also called context vector. The decoder input is an output encoder vector and start token, which characterizes the beginning of the generated sentence. The generated word depends on the previous decoder state and the last generated word.

3.5 Decoding strategies

Decoding strategies that optimize for output with high probability, such as **beam search**, make text incoherent and repetitive. A beam search is a limited-width breadth first search. This method starts from an empty sequence ($t = 0$), at every step $t = 0, 1, 2, 3, ...$ beam search expands at most k partial sequences (with highest probabilities) and computes the probabilities of sequences with length $t + 1$. It terminates with a beam of k complete sequences.

According to [16] decoding strategies with likelihood maximising lead to text that is degenerate, even when using state-of-the-art models. If the most likely word is always sampled, the model generates repetitive and overly generic text, like "I don't know", because it is a typical answer to any question.

Popular sampling methods for generation texts are based on sampling from the distribution. Temperature and top k sampling are the most popular methods to combat sampling from the tail.

Temperature sampling is inspired by statistical thermodynamics, where high temperature means low energy states are more likely encountered. In a probability model logits are divided by the temperature, before feeding them into softmax (Equation 3.9, where t is a temperature, $u_{1:|V|}$ are logits). Setting $t \in [0, 1)$ skews the distribution towards high probability events, which implicitly lowers the mass in the tail distribution. The temperature parameter controls the shape of distribution without sufficiently suppressing the unreliable tail.

$$p(x = V_l|x_{1:i-1}) = \frac{exp(u_l/t)}{\sum_{l'} exp(u'_l/t)} \tag{3.9}$$

Top-k sampling means sorting by probability and probabilities for anything below the token k are set to 0. But in some cases, there are few words to choose, there is a risk of generating bland or generic text, while if k is large the top-k vocabulary will include inappropriate candidates which will have their probability of being sampled increased by the renormalization. The top-k vocabulary $V^{(k)} \subset V$ (the set of size k) maximizes $p' = \sum_{x \in V^{(k)}} P(x|x_{1:i-1})$. The distribution is then re-scaled as in Equation 3.10.

$$P'(x|x_{1:i-1}) = \begin{cases} P(x|x_{1:i-1})/p' & \text{if } x \in V^{(k)} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

The research [16] shows how different natural distribution of human texts and the distribution of machine text produced from maximum likelihood decoding. To resolve this problem authors introduced **Nucleus Sampling**. The concept is that the vast majority of probabilities are concentrated in a small subset (*nucleus*) of the vocabulary that tends to vary from one to a few hundred candidates. Sampling from the top-p portion of the probability mass expands and contracts the candidate pool dynamically. Formally, given a distribution $P(x|x_{1:i-1})$, the top-p vocabulary $V^{(p)} \in V$ is defined to satisfy the condition in the Equation 3.11, where p is a pre-chosen threshold.

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geqslant p \tag{3.11}$$

3.6 Attention

A neural attention mechanism is inspired with the human visual attention mechanism. Visual attention is able to focus on a certain region of an image with "high resolution", while perceiving the surrounding image in "low resolution", and then adjusting the focal point over time. An example of the attention mechanism is represented in the Figure 3.6.

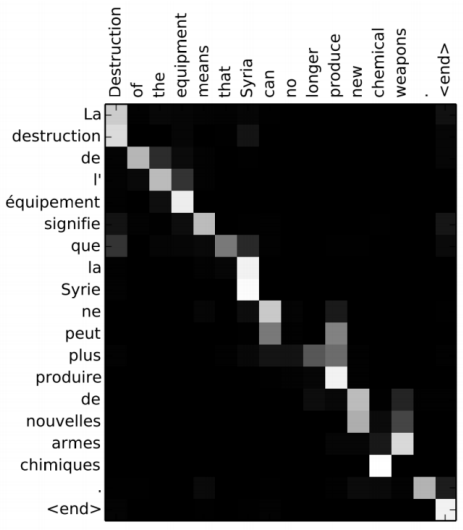

Figure 3.6: An example of attention mechanism. The x-axis and y-axis correspond to the words in the input sequence in English and the generated translation in French. Each pixel represents a weight.[2]

$$output_i = \sum_{j=1}^{T} \alpha_{ij} h_j \tag{3.12}$$

In the Equation 3.12 an output of the attention mechanism for time i is represented. T is a length of an input sequence, h_j is a hidden state representing a value, α_{ij} (Equation 3.13) is a weight of each annotation h_j.

[2] https://arxiv.org/pdf/1409.0473.pdf

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T} exp(e_{ik})} \tag{3.13}$$

$$e_{ij} = a(s_{i-1}, h_j) \tag{3.14}$$

In the Equation 3.14 e_{ij} represents an alignment model, represented as feedforward neural network a, which scores how well match the inputs around position j and the output at position i. s_{i-1} is RNN hidden state.

In [50] attention is described as "mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors". The output represents a computed weighted sum of the values (Equation 3.12). A weight for each value is computed by a compatibility function of the query with the corresponding key. Weights for keys corresponding to important values will be greater than for unimportant ones, which means that important values will be a bigger part of the output.

Self-attention is an attention mechanism, where "self" means that the inputs interact with each other and "attention" means that inputs find out who they should pay more attention. In the self-attention mechanism the query, keys and values are from the same sequence. The query is a single element from the sequence while the keys and values are the entire sequence. The attention output is a new representation of the element that was the query. Self-attention is used to compute a new representation of the sequence.

3.7 Transformer

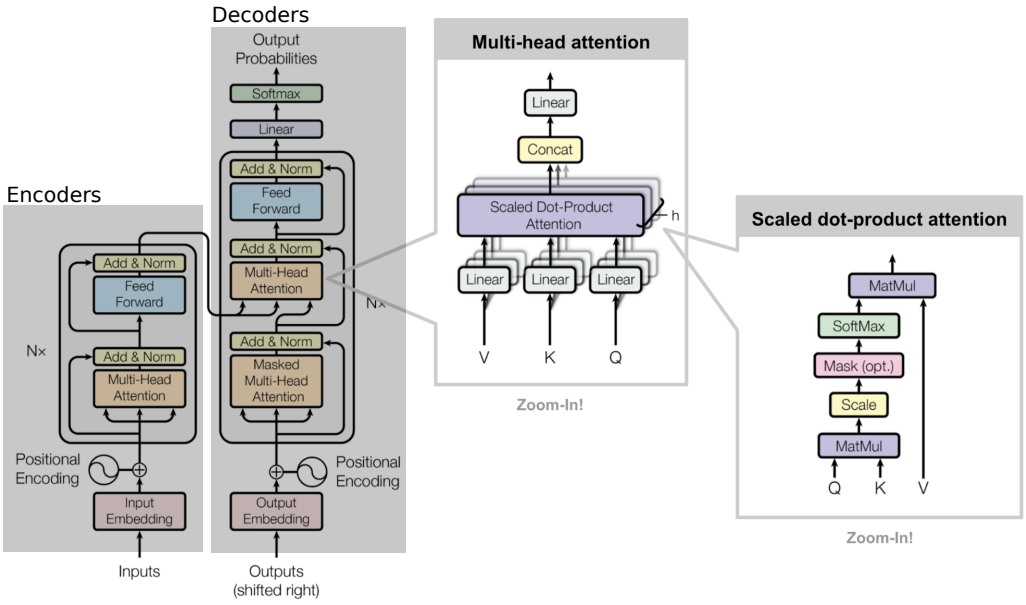

Figure 3.7: The architecture of the Transformer.[3]

The transformer [50] follows the seq2seq architecture that is based on self-attention mechanism. The model does not use any recurrent networks. In each step this model applies self-attention mechanism which directly models relationships between all words in

[3] http://primo.ai/index.php?title=Transformer

a sequence, regardless of their respective position. Transformers do not require that the sentence be processed in order, that allows process parallelization during training, unlike RNN.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.15}$$

$$MultiHead(Q, K, V) = Concat(head_1, head_2, ..., head_h)W^O \tag{3.16}$$

An Equation 3.15 represents a **scaled dot-product attention**. The input consists of values of dimension d_v, queries and keys of dimension d_k, where queries, keys and values are matricies.

An Equation 3.16 represents a **multi-head attention**, which allows the model to jointly track information from different representation subspaces at different positions. Averaging inhibits this with a single head of attention. The input also consists of queries, keys and values matricies, W^O is a parameter matrix, h is a number of parallel layers, $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$, where W_i^Q, W_i^K, W_i^V are parameter matricies. Q, K and V are different for each position of the attention modules in the structure. It depends on if they are in the encoder, the decoder or between them.

The architecture of the transformer is illustrated in the Figure 3.7. N_x in this Figure represents the number of encoder or decoder identical layers that can be stacked on top of each other multiple times. The inputs and output are first embedded into an n-dimensional space. Words' positions, n-dimensional vector, are added to the embedded representation of each word (Equation 3.17). These positions are used to inject some information about the relative or absolute position, because the transformer does not contain recurrence and convolution.

$$\begin{aligned} PE_{(pos,2i)} &= sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \\ PE_{(pos,2i+1)} &= cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \end{aligned} \tag{3.17}$$

Equation 3.17: The functions for calculating a word's positional encoding, where pos is a position, i is a dimension, d_{model} is the same dimension as the embeddings.

The transformer consists of a stack of encoders, where an encoder maps an input to a sequence of continuous representations. Each encoder has two sub-layers, where the first one is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. There is a residual connection [14] around each of the two sub-layers, followed by layer normalization. All sub-layers in the model produce outputs of dimension d_{model}, as well as the embedding layers.

The transformer also consists of a stack of decoders. In addition to the two sub-layers in each encoder, a decoder has an extra sub-layer, which performs multi-head attention over the output of the encoder stack. Masking in this layer ensures that the predictions for position k can depend only on the known outputs at positions less than k. There is also the residual connection.

3.8 Generative Pre-Training (GPT)

GPT [36] is a transformer-based (Section 3.7) language model. The model works in 2 stages. First of all transformer model trained on a very large amount of data in an unsupervised manner, then the model is fine-tuned on much smaller supervised dataset to help it solve specific tasks.

Unsupervised pre-training

In unsupervised pre-training a standard language modelling is used and the aim is to maximize the likelihood (Equation 3.18, where $U = u_1, u_2, ..., u_n$ is an unsupervised corpus of tokens, k is the size of context window, P is modeled using a neural network with parameter Θ).

$$L_1(U) = \sum_i \log P(u_i | u_{i-k}, ..., u_{i-1}; \Theta) \tag{3.18}$$

Multi-layer transformer decoder is used for the language model. The model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens (Equation 3.19), where $U = (u_{-k}, ..., u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, W_p is the position embedding matrix).

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= transformer\_block(h_{l-1}) \forall l \in [1, n] \\ P(u) &= softmax(h_n W_e^T) \end{aligned} \tag{3.19}$$

Supervised fine-tuning

After training the model, the parameters are adapted to the supervised target task. Given a labeled dataset C, where an instance represents a sequence of input tokens $(x^1, ..., x^m)$, along with a label y. The inputs are passed through the pre-trained model to get the final transformer block's activation h_l^m, which is then fed into an additional linear output layer with W_y parameters for predicting y (Equation 3.20).

$$P(y | x^1, ..., x^m) = softmax(h_l^{m\intercal} W_y) \tag{3.20}$$

$$L_2(C) = \sum_{(x,y)} \log P(y | x^1, ..., x^m) \tag{3.21}$$

Authors found that including language modelling as an auxiliary objective to the fine-tuning helped learning. First of all it improves the generalization of the supervised model. Secondly it accelerates convergence. They optimize the objective with weight λ (Equation 3.22).

$$L_3(C) = L_2(C) + \lambda * L_1(C) \tag{3.22}$$

3.9 BART

BART (Figure 3.8) is a denoising autoencoder for pre-training sequence-to-sequence models that maps a corrupted document to the original document it was derived from. The model

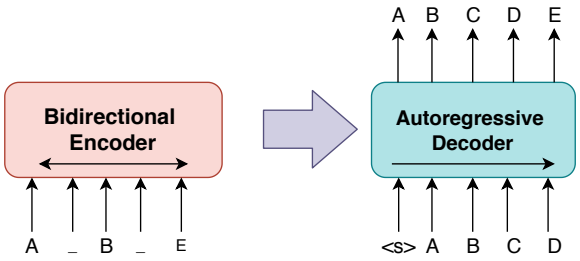

Figure 3.8: BART: Encoder inputs need not be aligned with decoder outputs, it allows arbitary noise transformations. A document is corrupted by replacing spans of text with mask symbols. The corrupted document is encoded with a bidirectional model, and then the likelihood of the original document is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and representations from the final hidden state of the decoder are used [23].

uses a standard transformer-based seq2seq architecture, with a bidirectional encoder and left-to-right decoder.

Bidirectional encoder consists of N transformer encoder blocks stacked on top of each other (Section 3.7), output is taken from the final block. The input is a sequence of tokens, which are first embedded into vectors and the processed in the model. The output is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index. Autoregressive decoder is GPT model (Section 3.8). Authors modified ReLU activation functions (Equation 3.23) to GELU (Equation 3.24) and initialized parameters from $\mathcal{N}(0, 0.2)$. Autoregressive decoder can be directly fine tuned for sequence generation tasks.

$$f(x) = x^{+} = max(0, x) \tag{3.23}$$

Equation 3.23: Rectified Linear Unit (ReLU).

$$GELU(x) = xP(X \leq x) = x\Phi(x) \tag{3.24}$$

Equation 3.24: Gaussian Error Linear Unit (GELU), where P is a Gaussian Probability Density Function.

Pre-training has two stages: text corruption with an arbitrary noising function and learning to reconstruct the original text. BART allows to apply any type of document corruption, such as token masking, sentence permutation, document rotation, token deletion, text infilling (Figure 3.9).

In **Token Masking** part of words in each sequence are replaced with a `[MASK]` token before embedding. The model then tries to predict the original value of the masked words, based on the context provided non-masked words in the sequence. Bidirectional encoder loss function takes into consideration only the prediction of masked values. Pre-training on this task allows the model to learn general features of the language.

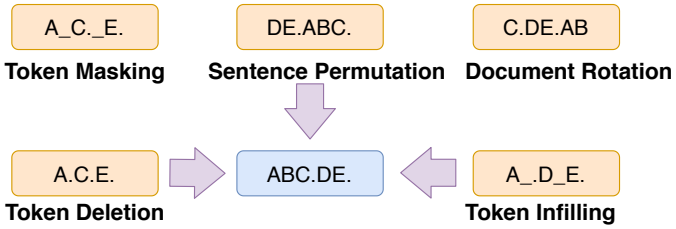

Figure 3.9: Corruptions for noising the input. These transformations can be composed.

In **Token Deletion** random tokens are deleted from the input. It is needed to decide which positions are missing inputs.

For **Text Infilling** a number of text spans are sampled. Span lengths represent a Poisson distribution ($\lambda = 3$) and each span is replaced with a single `[MASK]` token. The model learns to predict how many tokens are missing from a span.

In **Document Rotation** a token is chosen uniformly at random. A document rotation starts with that token. The model learns to identify the start of the document.

For **Sentence Permutation** a document is divided into sentences based on full stops (full stop is a punctuation marking the end of a declarative sentence). These sentences are randomly shuffled.

Chapter 4

Related work

This chapter presents an overview of the most popular NLG models for building open-ended dialogue systems.

Modelling conversations with generative probabilistic models was first proposed in [37]. They present a data-driven approach to generating responses to Twitter status posts, based on phrase-based Statistical Machine Translation. In [43] authors proposed a framework for generating responses on micro-blogging websites with using recurrent neural networks. Their model can generate grammatically correct and content-wise appropriate responses to over 75% of the input text.

The idea that computers can generate stylized texts already appeared half a century ago in [52]. Authors choose a variety of text characteristics as style, such as an expession of politeness in machine translation [42], transformation from modern English to Shakespearean English [17], sentiment of a text in [44] and [24]. In [10] authors used a structured latent space to generate stylized dialogue responses. Another approach was proposed in [18], where authors applied an adversarial loss to separate style from content. In [48] authors describe the problem of stylized text generation in a multilingual setup and show the importance of phonetics for generating the author's stylized poetry.

4.1 Attributes control

In [41] solutions to common NLG problems in dialogue systems are described. Authors add control (the ability to specify desired attributes of the generated text at test time) and focus on four controllable attributes of text: repetition, specificity, response-relatedness and question-asking. They measure repetitiveness as n-gram overlap, specificity as word rareness, response-relatedness as the embedding similarity of the bot's response to the human's last sequence. In this work, authors use **Conditional Training (CT)** [34] and **Weighted Decoding (WD)** [11].

A **CT** model learns probabilities $P(y|x,z)$, where y is the output text, x is the input text and z is a control variable, which specifies the desired output attribute. In the model z is presented with learned embedding and is concatenated to each decoder input. For example, to get very generic or very specific response, z can be set to an embedding representing `<LOW>` or `<HIGH>` (Figure 4.1). If it is necessary simultaneously control several attributes, multiple control embeddings $(z_1, z_2, ..., z_n)$ can be concatenated and the model learns $P(y|x, z_1, z_2, ..., z_n)$. Disadvantage of Conditional Training is that it can't control

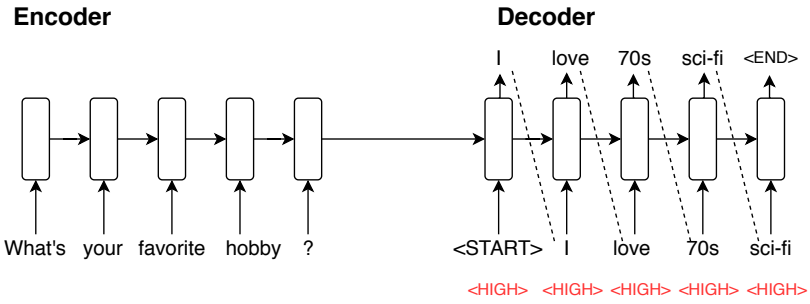

Figure 4.1: Example of Condition Training model.

attributes without sufficient training data. Another observed flaw is that the CT model learns only a very weak connection between z and the semantic relatedness of the output.

A **WD** is a technique applied during decoding to increase/decrease the probability of words with certain features.

$$score(w, y_{<t}; x) = score(y_{<t}; x) + \log P_{RNN}(w|y_{<t}, x) + \sum_i w_i * f_i(w; y_{<t}, x) \tag{4.1}$$

In weighted decoding (Equation 4.1), a hypothesis $y_{<t} = y_1, ..., y_{t-1}$ is expanded by computing the score for each possible next word w in the vocabulary on the t^{th} step of decoding. $\log P_{RNN}(w|y_{<t}, x)$ is the log-probability of the word w assigned by the probabilistic model. $score(y_{<t}; x)$ is the accumulated score of the already-generated words in the hypothesis $y_{<t}$. $f_i(w; y_{<t}, x)$ is a decoding feature with associated weights w_i (hyperparameters to be chosen). Each feature presents a specific controlling attribute.

$$NIDF(w) = \frac{IDF(w) - min\_idf}{max\_idf - min\_idf} \tag{4.2}$$

Normalized Inverse Document Frequency (NIDF) is used as a measure of word rareness. In conditional training the metric is represented as z variable and in weighted decoding as a parameter of rareness for a word prediction (On each step of the decoding, the probability of each word in the vocabulary is updated in proportion to its rareness. The size of the update is controlled by a weight parameter.) (Equation 4.2). $IDF(w) = \log(\frac{R}{c_w})$ is a Inverse Document Frequency of a word w, where R is the number of responses in the dataset, c_w is the number of those responses that contain w. min_idf and max_idf are the minimum and maximum IDF's, which are used for normalising the NIDF (ranges from 0 to 1).

4.2 Emotional Chatting Machine (ECM)

Emotional Chatting Machine (ECM) [54] can generate not only relevant and grammatical responses, but also emotionally consistent. This framework proposes seq2seq architecture. This separate responses into several categories (Angry, Disgust, Happy, Like, Sad, Other). The emotion category of the to-be-generated response is given for ECM, because, according to the authors, emotions are highly subjective.

In the architecture (Figure 4.2) a post can be answered with different emotions, depending on the attitude of the respondent. For example, for a sad story, someone may respond

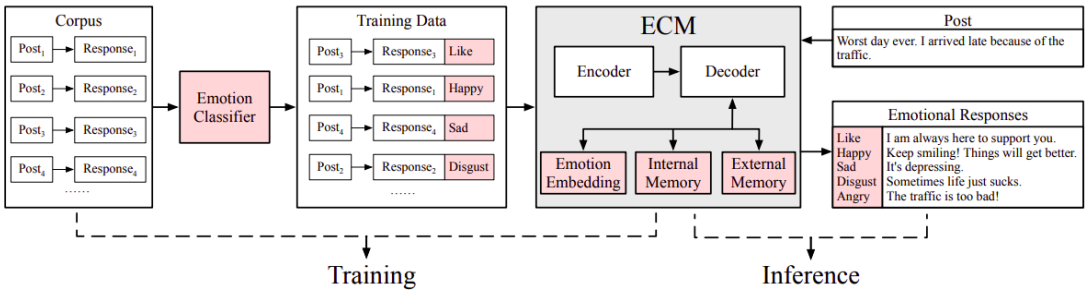

Figure 4.2: Overview of ECM (the grey unit). The pink units are used to model emotion factors in the framework.

with anger (as an irritable stranger), sympathy (as a friend) or happy (as an enemy). *Emotion classifier* generates labels for each response. Generated labels and responses are fed into ECM to generate emotional responses conditioned on different emotion categories.

$$s_t = GRU(s_{t-1}, [c_t; e(y_{t-1}); v_e]) \tag{4.3}$$

Equation 4.3: ECM decoder's state. GRU [7] is a gating mechanism in RNN.

In *Emotion Category Embedding* (Equation 4.3) each emotion category of a response is represented by a real-valued, low dimensional vector. The vector of an emotion category v_e for each category e is randomly initialize. During the training the model is learning the vectors of the emotion category. The emotion category embedding, along with the context vector c_t, and the word embedding $e(y_{t-1})$, are fed into the decoder to update the decoder's state s_t.

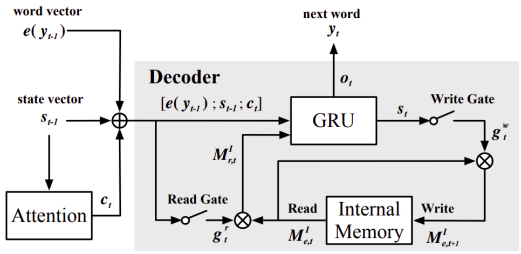

Figure 4.3: Data flow of the decoder with an internal memory.

$$g_t^r = \sigma(W_g^r[e(y_{t-1}); s_{t-1}; c_t]) \tag{4.4}$$

$$g_t^w = \sigma(W_g^w s_t) \tag{4.5}$$

Internal memory captures emotion dynamics during decoding. It is achieved using the following concept: before the decoding process each category has an internal emotion state; at each step the emotion state decays by a certain amount; when decoding process is

completed, the emotion state should be zero, what indicates that the emotion is completely expressed. The detailed process is represented in the Figure 4.3, where at each time step t, the gate g_t^r (Equation 4.4) is computed with the input of the word embedding of the previously decoded word $e(y_{t-1})$, the previous state vector of the decoder s_{t-1}, and the current context vector c_t. The write gate g_t^w (Equation 4.5) is computed on the decoder's state s_t. These gates are used to read from and write into the internal memory.

$$M_{r,t}^I = g_t^r \odot M_{e,t}^I \tag{4.6}$$

$$M_{e,t+1}^I = g_t^w \odot M_{e,t}^I \tag{4.7}$$

The emotion state is erased by a g_t^w at each time step. At the last step, the internal emotion state will decay to zero. This process is represented in the Equations 4.6, 4.7, where M means memory, I denotes Internal, r/w denotes read and write respectively, $\odot$ is element-wise multiplication.

$$s_t = GRU(s_{t-1}, [c_t; e(y_{t-1}); M_{r,t}^I]) \tag{4.8}$$

The decoder's state s_t is conditioned on the previous state of the decoder s_{t-1}, the context vector c_t, the previous target word $e(y_{t-1})$ and the emotion state update $M_{r,t}^I$ (Equation 4.8).

External memory is used to model emotion expressions explicitly, because in the internal memory module the correlation between the change of the internal emotion state and selection of a word is not directly observable. The model can choose to generate words from an emotion vocabulary or a generic vocabulary.

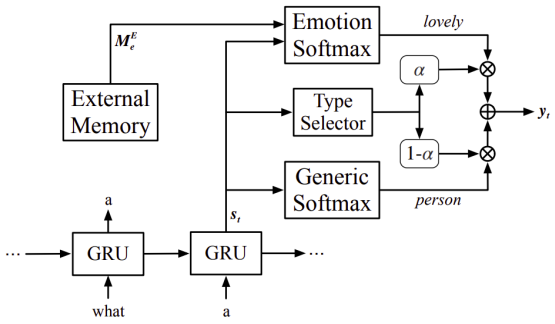

Figure 4.4: Data flow of the decoder with an external memory.

$$\alpha_t = \sigma(v_u^\intercal s_t) \tag{4.9}$$

$$P_g(y_t = w_g) = softmax(W_g^o s_t) \tag{4.10}$$

$$P_e(y_t = w_e) = softmax(W_e^o s_t) \tag{4.11}$$

$$y_t \sim o_t = P(y_t) = \begin{cases} (1-\alpha_t)P_g(y_t = w_g) \\ \alpha_t P_e(y_t = w_e) \end{cases} \tag{4.12}$$

The decoder with an external memory is represented in the Figure 4.4. The current decoder's state s_t, the emotion softmax (Equation 4.10) and the generic softmax (Equation 4.11) are computed over the emotion vocabulary. This vocabulary is read from the external memory and the generic vocabulary. α_t (Equation 4.9) is the type selector, which controls the weight of generating an emotion or a generic word. The next word y_t (Equation 4.12) is sampled from the concatenation of the two weighted probabilities. α_t is in range $[0, 1]$, and it is used to balance the choice between an emotion word w_e and a generic word w_g. P_g and P_e in the Equation 4.12 are the distributions over generic and emotion words respectively. Two vocabularies have no intersection, and the final word decoding distribution $P(y_t)$ is a concatenation of two distributions.

In ECM three mechanisms were proposed to model the emotion factor. An important flaw of ECM is that the model needs an external decision maker, because this model has to specify an emotion category that should be generated.

Chapter 5

Evaluation and datasets

Corpus is very important for successful Natural Language Generation. Dialogue systems require training data in the format of people text conversation, for example, non-fiction or movie reviews are not suitable for this. Large volumes of training data improves the decision-making ability of NLG model, so those models can use it to figure out patterns. Unfortunately, there are not a lot of datasets available for training NLG models, probably due to the high cost of creating quality datasets.

5.1 Persona-Chat dataset

Persona 1	**Persona 2**
I like to ski.	I am an artist.
My wife does not like me anymore.	I have four children.
I have went to Mexico 4 times this year.	I recently got a car.
I hate Mexican food.	I enjoy walking for exercise.
I like to eat cheetos.	I love watching Game of Thrones.
[PERSON 1:] Hi	
[PERSON 2:] Hello! How are you today?	
[PERSON 1:] I am good thank you, how are you.	
[PERSON 2:] Great, thanks! My children and I were just about to watch Game of Thrones.	
[PERSON 1:] Nice! How old are your children?	
[PERSON 2:] I have four that range in age from 10 to 21. You?	
[PERSON 1:] I do not have children at the moment.	
[PERSON 2:] That just means you get to keep all the popcorn for yourself.	
[PERSON 1:] And Cheetos at the moment!	
[PERSON 2:] Good choice. Do you watch Game of Thrones?	
[PERSON 1:] No, I do not have much time for TV.	
[PERSON 2:] I usually spend my time painting: but, I love the show.	

Table 5.1: Example of a dialogue from the Persona-Chat dataset [53].

Persona-Chat models normal conversation when 2 people meet for the first meet and try to get know each other better. The aim of the dialogue is to learn about interests of another person, find common ground and discuss their hobbies. The task involves both asking and answering questions.

Original Persona	Revised Persona
I love the beach.	For me, there is nothing like a day at the seashore.
My dad has a car dealership.	My father sales vehicles for a living.
I just got my nails done.	I love to pamper myself on a regular basis.
I am on a diet now.	I need to lose weight.
Horses are my favorite animal.	I am into equastrian sports.

Table 5.2: Example of original and revised personas [53].

Average number of words in first persona description:	6.332
Average number of words in second persona description:	6.321
Average number of words in the first person's sequences:	11.419
Average number of words in the second person's sequences:	11.929
Number of first persona description's sentences:	40239
Number of second persona description's sentences:	40126
Number of the first person's sequences:	65719
Number of the second person's sequences:	65719
Number of dialogues	8938

Table 5.3: Persona-Chat statistics.

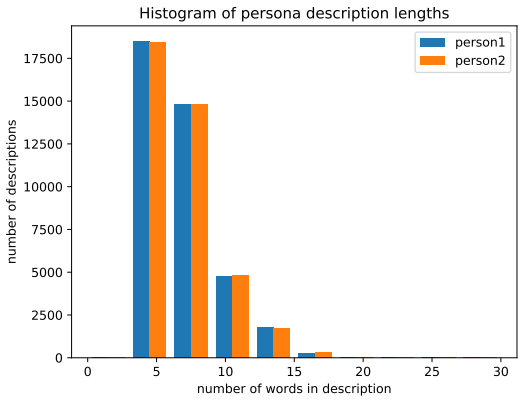

Figure 5.1: Histogram of persona description lengths.

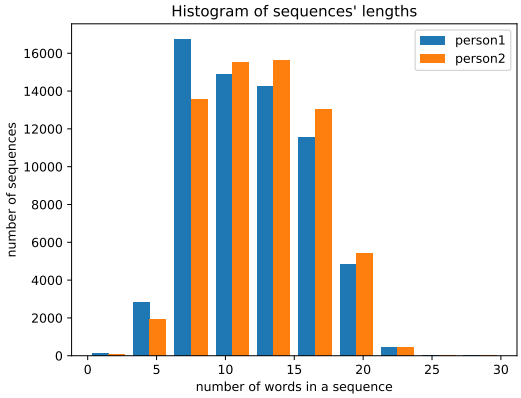

Figure 5.2: Histogram of sequences' lengths of a person.

Persona-Chat dataset consists of small conversations between 2 crowdworkers from Amazon Mechanical Turk who were randomly paired and asked to act the part of a given provided persona (randomly assigned, and created by another set of crowdworkers). The data collection consists of persona chat (each dialogue has 6-8 turns), personas (set of 1155 possible personas, each consisting of at least 5 profile sentences), revised personas to avoid word overlap, because crowdworkers sometimes could repeat profile information in a chat (the Table 5.2). In turn-based dialogue each message consists of a maximum of 15 words. All statistics are presented in the Table 5.3, the Figure 5.1 and the Figure 5.2. An example of Persona-Chat dialogue is shown in the Table 5.1. The dataset contains 262,848 message-responses pairs.

5.2 Datasets for stylisation and pre-training

Stanford Sentiment Treebank (SST) contains sentences from movie reviews and includes labels for every syntactically plausible phrase in thousands of sentences. This corpus represents sentiment of reviews. To create SST the corpus of movie review excerpts from the paper [32] is used, where HTML tags and sentences that are not in English are deleted from this dataset. The Stanford Parser is used to parse all 10,662 sentences and some snippet were splitted into multiple sentences. The resulting 215,154 phrases were labeled by Amazon Mechanical Trunk. There are 25 different labels from *very negative* to *very positive*. In this thesis negative and somewhat negative classes are used as one negative class and positive, somewhat positive classes as one positive class. A histogram for positive and negative classes is represented in the Figure 5.3.

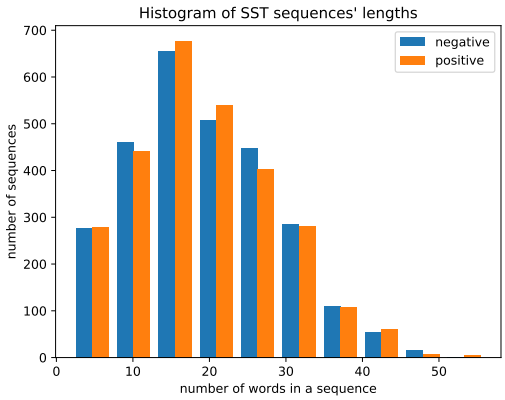

Figure 5.3: Histogram of Stanford Sentiment Treebank labeled reviews.

For poetic style the **Shakespeare** dataset was used, which contains all of Shakespeare's plays. A histogram for this dataset is shown in the Figure 5.4.

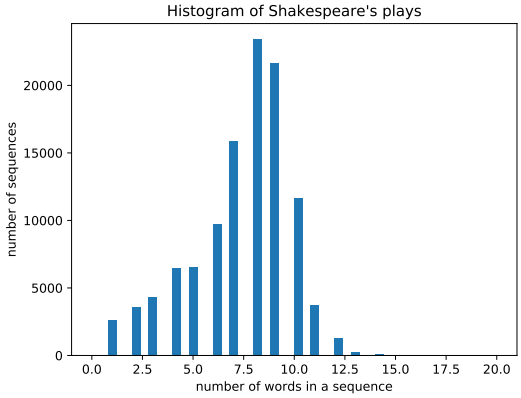

Figure 5.4: Histogram of Shakespeare's plays.

A dataset of **english plaintext jokes** was used for generating text with humor style. There are about 208 000 jokes scraped from 3 sources (Table 5.4). stupidstuff.json is scrapped from stupidstuff.org, wocka.jsom from `http://wocka.com`, reddit_jokes.json is scraped from `https://www.reddit.com/r/Jokes` and contains all submissions to the subreddit as of 13.02.2017. A histogram for the dataset is shown in the Figure 5.5.

reddit_jokes.json	195K jokes	7.40M tokens
stupidstuff.json	3.77K jokes	396K tokens
wocka.json	10.0K jokes	1.11M tokens
TOTAL	208K jokes	8.91M tokens

Table 5.4: Statistics of jokes' dataset.

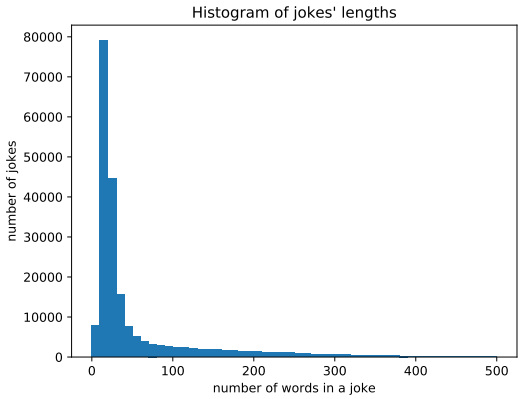

Figure 5.5: Histogram of jokes lengths.

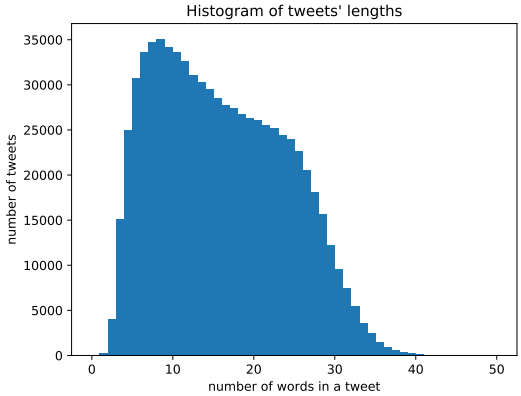

Figure 5.6: Histogram of tweets' lengths.

Twitter dataset contains 867,710 message-response pairs from Twitter and is used for pre-training baseline model. A histogram for the dataset is shown in the Figure 5.6.

5.3 Evaluation methods

The majority of NLG researches published between 2005-2014 relies on automatic metrics [13]. A lot of NLG evaluations use automatic metrics, mostly because it is a cheap and fast method. These metrics are reasonable if they are known to be enough correlated with human preferences. According to [2] conversational dialogue systems cannot be evaluated by automatic metrics, because dialogue is heavily dependent on context and theory of current dialogue is not precise enough to predetermine the target output.

In [30] the weak correlation between human and automatic evaluations is also confirmed. Authors compared different word-based (relies on ground-truth references) and grammar-based metrics (does not rely on ground-truth references). Their model, combination of WBMs and GBMs, achieved high correlation with humans but only within a single domain.

Word-based metrics METEOR [3], BLEU [33], ROUGE [25] are usually used for automatic summarization. They assume that valid responses have significant word overlap with the ground truth responses. In dialogue systems these metrics cannot be used, because there is significant diversity in the space of valid responses to a given context [26].

Grammar-based metrics have been explored in machine translation [12] or grammatical error correction [29]. The Flesch Reading Ease score [9] calculates a ratio between the number of characters per sentence, the number of words per sentence, and the number of syllables per word. Designed grammar-scoring functions are presented in the [29]. These metrics can be easily manipulated with grammatically correct and easily readable output that is unrelated to the input.

Best practices for the human evaluation of NLG are represented in [21]. According to these practices it is better to use separate criteria rather than an overall quality assessment. The effect of fatigue (annotators can change their responses due to fatigue) can be reduced by shortening the task. Order of the compared models may also affect annotators. To reduce this effect, presenting the conditions should be in a systematically varied order. There should always be quite a few participants.

A/B testing [20] is used as human evaluation. This method is described as the randomized assignment to the user of two variants: the *control* and the *treatment*. The control

variant typically is the existing version, which in this case it is a real dialogue of two people. The treatment variant is the new version being evaluated – in this case it is a generated dialogue.

$$p_X(k) = Pr(X = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}} \tag{5.1}$$

Equation 5.1: Hypergeometric distribution, where N is the population size, K is the number of success states in the population, n is the quantity drawn in each trial, k is the number of observed successes. $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ is a binomial coefficient.

Fisher's exact test [8] is carried out to check human evaluation for statistically significant biases in it. This test is used when there are two nominal variables (nominal variables classify observations into discrete categories) to know if the values of one variable differ among the values of another variable. In this case it is controlled if the order of the dialogs affects the user. Fisher's exact test is used when the total sample size is less than 1000. The null hypothesis is that the one variable is independent of the second variable. To confirm or refute this hypothesis, it is necessary to calculate the probability of getting the observed data, and all data sets with more extreme deviations, under the null hypothesis. Hypergeometric distribution (Equation 5.1) is used to calculate the probability. If the probability is less than 5%, the null hypothesis is rejected.

Chapter 6

Design and implementation

This chapter describes the model for stylized text generation in dialogue systems and performed experiments. All experiments can be divided into 3 parts: decoding strategies, feature-based decoding modifications and weighted decoding of a combination of language models. The designed architecture for WD of a language models combination is represented in the Figure 6.1 and implemented in two variants. First variant uses a LSTM-based baseline model. The second one uses pre-trained transformer-based models. All parts of the architecture were implemented in Python 3.7.3 with using PyTorch 1.4.0 framework.

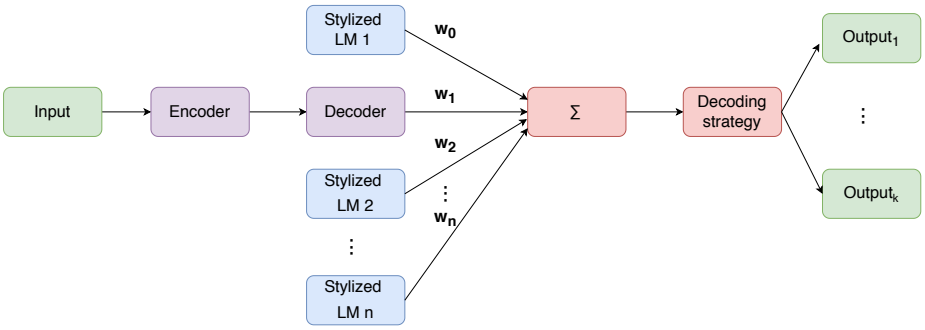

Figure 6.1: An architecture of combination models for stylized Natural Language Generation. w_n is a hyperparameter.

An experiment with decoding strategies uses Beam search and Nucleus sampling methods. These decoding strategies, described in detail in the section 3.5, allow the generation of less generic and repetitive sequences, which affects the user's perception of the text.

For feature-based decoding modifications only encoder-decoder model is used. NIDF metric, described in the chapter 4, is used as feature-based modification. This approach makes generated text more specific. NIDF represents a measure of word rareness and on each step of the decoding, the probability of each word in the vocabulary is updated in proportion to its rareness. This metric represents a weight for each word probability from the encoder-decoder model. Another experimet is with a length of generated sequences, because a person's style is represented not only by words, but also by the length of sentences. Some people are used to writing very briefly, and someone likes to paint every detail in a text. With each iteration, when generating the next word, a small weight is added to the probability of the `[EOS]` (end of sequence) symbol. The last experiment with feature-based decoding modification is blocking stop words. During the decoding useless data are filtered.

In NLP useless data are referred to as stop words (a commonly used words, for example, “a”, “the”, “I” etc.). This approach could help to generate more unusual sequences, for example, instead of “I think” to use the phrase “In my opinion”. Stop words are used from NLTK[1](suite of libraries and programs for symbolic and statistical NLP). During decoding in each iteration, when selecting the next word in the sequence, it is checked if this word is not contained in stop words. If word is in stop words, a probability for this word is set to 0.

$$p(y_1, ..., y_{T'}|x_1, ..., x_T) \propto \prod_{t=1}^{T'}[w_1 * p_1(y_t|v, y_1, ..., y_{t-1}) + \sum_{i=2}^{n} w_i * p_i(y_t|y_1, ..., y_{t-1})] \quad (6.1)$$

The Equation 6.1 is a formal description of the experiment with weighted decoding (Figure 6.1), where $x_1, ..., x_T$ is an input sequence and $y_1, ..., y_{T'}$ is its corresponding output sequence whose length T' may differ from T. $p_1(y_t|v, y_1, ..., y_{t-1})$ is a distribution for encoder-decoder language model, in which v is the last hidden state of the encoder. $p_i(y_t|v, y_1, ..., y_{t-1})$ is a distribution for each stylized language model, n is a number of stylized models. All distributions are multiplied by a corresponding weight w, a hyperparameter in range [0,1] to be chosen.

Last experiment represents switching models, where n words in a sequence are generated over encoder-decoder model and n words are generated over stylized language model, n is a hyperparameter. Sequence generation stops when the end mark of the sequence is generated or the length of the sequence is equal to the maximum length.

For training language models several datasets are splitted. Persona-Chat is splitted into train data – 70%, valid data – 20% and test data – 10%. For SST dataset annotators most often used only 5-class classification: negative, somewhat negative, neutral, positive or somewhat positive. Many of sentences could be considered neutral, therefore, to generate text with positive and negative sentiment I have used negative and somewhat negative classes as one negative class and positive, somewhat positive classes as one positive class. Twitter dataset is splitted into train data – 85%, valid data – 10% and test data – 5%.

6.1 Baseline

Encoder-decoder model in baseline is implemented as LSTM sequence-to-sequence language model, represented in the chapter 3, with Luong attention [27]. This model is pre-trained on message-response pairs from Twitter dataset (subsection 5.2) and trained on the Persona-Chat dataset, described in the subsection 5.1. As the input x to the encoder-decoder model, the entire dialogue history, separated by unique token, is used.

Stylized language model is implemented as LSTM-based RNN model. Each stylized language model is trained separately on the corresponding stylistic dataset described in the subsection 5.2.

Data are tokenized, where tokenization is a process of splitting text into units represented at model’s input and replacing sensitive data with unique identification symbols that retain all the essential information about the data without compromising its security. Spacy[2]library is used with few handcrafted rules for tokenization. As input GloVe (Global Vectors) embeddings are used for distributed word representation [35]. It is a model, which

[1] https://www.nltk.org/

maps words into a meaningful space where the distance between words is related to semantic similarity. The model was trained only on the nonzero elements in a word-word cooccurrence matrix. Outputs of encoder-decoder and stylized language models (logarithmic probabilities of words) are multiplied by weights (weight is a hyperparameter to be chosen) to increase or decrease the probability of words and finally these outputs are added up (Figure 6.2).

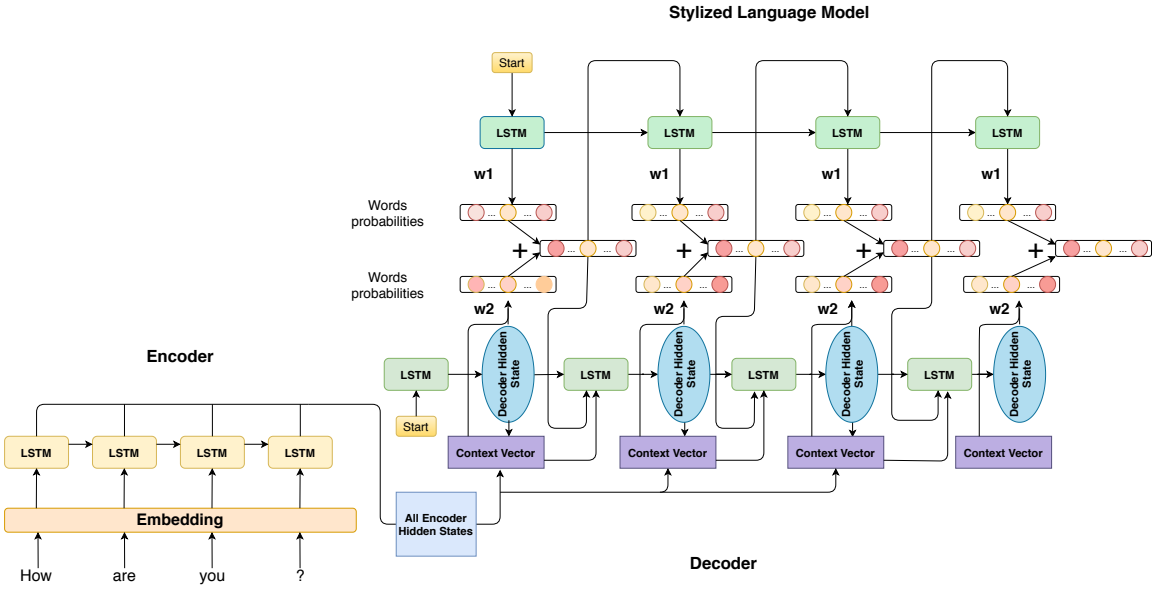

Figure 6.2: Stylized Natural Language Generation baseline model.

Luong attention model is classified into 2 categories, *global* and *local*. Common to these types of model is the fact that at each time step t in the decoding phase previous hidden state is taken as input to derive a context vector $\mathbf{c_t}$, that captures relevant information to predict the current target word y_t. This categories differ only if "attention" is placed on all source positions or on a few source positions.

The simple concatenation layer combines the information from vectors h_t and c_t to produce an attentional hidden state (Equation 6.2).

$$\widetilde{\mathbf{h_t}} = tanh(\mathbf{W_c}[\mathbf{c_t}; \mathbf{h_t}]) \tag{6.2}$$

The attention vector $\widetilde{\mathbf{h_t}}$ then passed through the softmax layer to produce the predictive distribution (Equation 6.3).

$$p(y_t|y_{<t}, x) = softmax(\mathbf{W_s}\widetilde{\mathbf{h_t}}) \tag{6.3}$$

Global Attention

An alignment vector a_t (size of a_t is equal to the number of time steps on the source side) is derived by comparing the current target hidden state $\mathbf{h_t}$ with each source hidden state $\bar{\mathbf{h}}_\mathbf{s}$ (Equation 6.4).

$$a_t(s) = align(\mathbf{h_t}, \bar{\mathbf{h}}_\mathbf{s}) = \frac{exp(score(\mathbf{h_t}, \bar{\mathbf{h}}_\mathbf{s}))}{\sum_{s'} exp(score(\mathbf{h_t}, \bar{\mathbf{h}}_\mathbf{s}))} \tag{6.4}$$

[2] https://spacy.io/

There are three types of the score function (the score function is referred as a content-based function) (Equation 6.5).

$$score(\mathbf{h_t}, \bar{\mathbf{h}}_\mathbf{s}) = \begin{cases} \mathbf{h_t}^\intercal \bar{\mathbf{h}}_\mathbf{s}, & \text{dot} \\ \mathbf{h_t}^\intercal \mathbf{W_a} \bar{\mathbf{h}}_\mathbf{s}, & \text{general} \\ \mathbf{v_a}^\intercal tanh(\mathbf{W_a}[\mathbf{h_t}; \bar{\mathbf{h}}_\mathbf{s}]), & \text{concat} \end{cases} \tag{6.5}$$

In location-based function the alignment scores are computed from solely the target hidden state $\mathbf{h_t}$ (Equation 6.6).

$$a_t = softmax(\mathbf{W_a h_t}) \tag{6.6}$$

The context vector $\mathbf{c_t}$ is computed as the weighted average over all the source hidden state, where alignment vector represents weights.

Local Attention

Global attention is expensive, because it has to attend to all words on the source side for each target word. Local attention chooses to focus only on a small subset of the source positions per target word.

The local alignment vector a_t in this category of attention is fixed-dimensional, because of it there are 2 variants of the model, *monotonic* (Equation 6.7) and *predictive* (Equation 6.8).

$$p_t = t \tag{6.7}$$

$$p_t = S \cdot sigmoid(\mathbf{v_p}^\intercal tanh(\mathbf{W_p h_t})) \tag{6.8}$$

In monotonic alignment the source and target sequences are roughly monotonically aligned. In predictive alignment the model learns to predict the alignment position, where $\mathbf{W_p}$ and $\mathbf{v_p}$ are the learned model parameters.

Gaussian distribution centered in p_t is used to favor alignment points near p_t (Equation 6.9).

$$a_t(s) = align(\mathbf{h_t}, \bar{\mathbf{h}}_\mathbf{s}) exp(-\frac{(s - p_t)^2}{2\sigma^2}) \tag{6.9}$$

$$H(p, q) = -\sum_{x \in X} p(x) \log q(x) \tag{6.10}$$

Equation 6.10: p is the target distribution, q is the approximation of the target distribution. $p(x)$ is the probability of the event x in p, $q(x)$ is the probability of the event x in q.

For all experiments with the baseline 2 hidden layers were used in both the encoder and decoder with 512 hidden unites in each hidden layer. A dropout [45] was used with the probability of 50% to prevent neural networks from overfitting. *Cross-entropy* (measure of the difference between 2 probability distributions for a given random variable or set of events) was used as a loss function. Formally representation of cross-entropy is in the Equation 6.10. *Adam* [19] is used as the optimization algorithm.

6.2 Pre-trained models

Combination of BART and GPT-2 models is used to improve results of the created architecture (Figure 6.1). BART is an autoencoder for pre-training sequence-to-sequence models, described in the section 3.9. BART model with 400M hyperparameters has 12 encoder and decoder layers. This model is chosen because it achieves state-of-the-art results on Conv AI2 task (a dialogue response generation task, conditioned on context and a persona). GPT-2 is a large transformer-based language model, which was trained to predict the next word in 40GB of Internet text. GPT-2 with 117M hyperparameters has large improvements on WikiText2 dataset for language modelling. This model is a successor to GPT, described in the section 3.8 and used as stylized language model.

Model name	**Valid F1**	**Valid PPL**
Seq2Seq + Attention	16.02	35.07
BART	20.72	11.85

Table 6.1: BART outperforms previous work on conversational response generation. Results are from [23]. **PPL** (perplexity) is a measurement of how well a probability model predicts a sample. A low perplexity is good at predicting the sample, because perplexity is just an exponentiation of the entropy. **F1** is a measure of a test's accuracy. $F1 = \frac{2}{recall^{-1}+precision^{-1}}$, where *recall* is the number of correct positive results divided by all samples that should have been identified as positive and *precision* is the number of correct positive results divided by the number of all positive results returned by the model.

BART decoder is an equivalent to the GPT model. GPT-2 is a direct scale-up of GPT, with more parameters (1.5 billion parameters) and trained on more amount of data. GPT-2 model uses more words in a vocabulary than GPT, what means that before a combination of these models, it is necessary to convert a distribution of probabilities of the next word in a sequene over all the words in the smallest vocabulary to the distribution over the biggest one and add 0 if word does not exist. Outputs of BART model, transformed to GPT-2 vocabulary, and logarithmic probabilities of words of GPT-2 models are multiplied by weights to increase or decrease the probability of words and finally these outputs are added up.

Each GPT-2 model represents one style. 4 such models are fine-tuned for humor, positive and negative sentiment and poetic style. The model is trained with batch size parameter set to 1, learning rate for Adam is 1e−4, 50000 chars is a minimum size to concatenate input files with `<|endoftext|>` separator.

BART is fine-tuned on Persona-Chat dataset. Learning rate is set to 3e−5, label smoothing is 0.1, dropout is also 0.1.

6.3 Proposed evaluation approach

There is a lot of research, which shows that human and automatic evaluations have weak correlation (Subsection 5.3). In dialogue systems there are lots of ways how to answer the opponent's correctly, that's why in this thesis only human evaluaton is used.

A/B tests were used for human evaluation. Several manifestations of human styles were evaluated, such as sentiment (positive, negative), poetry, humor, and specificity. Users were given a questionnaire with several questions ("Which dialogue is more positive?", "Which dialogue is more negative?", "Which dialogue is more poetic?", etc.) and for each question

there were 2 dialogs as answer options. One of the dialogs was generated by the model, and the second was from real life. In the assessment, 21 people took part in the age range from 1991 to 2004 with different levels of education and field of activity.

For the Fisher's exact test, 2 questionaire were created, where there was only 1 question ("Which dialogue is more poetic?"). In the first case, as a response option, the generated dialogue was first given and then the real dialogue, and in the second, vice versa.

Chapter 7

Results and discussion

In this chapter, the results of the experiments described in the previous section are presented and their implications are discussed. Human evaluation occurred only for those experiments that were subjectively considered more successful.

All experiments with features, decoding strategies and combination of models are carried out on pre-trained models, because the dialogs generated by the baseline model do not make sense in the context. The example in the Table 7.1 shows that the model also has a repetition problem, such as repetition of the part of an sequence ("I am"), repetition of phrases ("I am afraid"). These problems can be because of the small amount of data and LSTM neural networks still have a problem to hold a context. In comparison with the baseline model, the dialogue generated using the BART model (Table 7.2) is more logical and diverse.

Person 1 description: 1. as a child, i won a national spelling bee. 2. i have been published in the new yorker magazine. 3. i am a gourmet cook. 4. i have perfect pitch.
Person 2 description: 1. i am very athletic. 2. i wear contacts. 3. i have brown hair. 4. i love bicycling. 5. i hate carrots.
Person1: i am afraid. **Person2:** i am a business employee. **Person1:** i am a surfer. **Person2:** i am not much of a foodie. **Person1:** i am afraid. **Person2:** that is cool. **Person1:** i do. i like to starve. **Person2:** i like to go to the homeless. **Person1:** i am not really a fan of harry potter. **Person2:** i am good. i am in tampa. **Person1:** i am from boston ma. **Person2:** i am in boston.

Table 7.1: Example of a dialogue conversation generated by baseline model.

Results of using beam search and Nucleus sampling methods are shown in the Tables 7.3 and 7.4. All these decoding strategies are described in detail in the section 3.5.

In Nucleus sampling cumulative distribution function (CDF) is computed and cut off as soon as CDF exceeds P. Determined the set of tokens, to be sampled from, may be very

Person 1 description: 1. as a child, i won a national spelling bee. 2. i have been published in the new yorker magazine. 3. i am a gourmet cook. 4. i have perfect pitch.
Person 2 description: 1. i am very athletic. 2. i wear contacts. 3. i have brown hair. 4. i love bicycling. 5. i hate carrots.
Person1: hi, how are you doing today? **Person2:** that is cool. I am a chef **Person1:** i don't like carrots either. I am a gourmet cook. **Person2:** i am very athletic. i love to play sports **Person1:** i ride my bike to work as well. **Person2:** that is awesome. i am very athletic, so i love spelling **Person1:** i have been published in a new yorker magazine **Person2:** that is cool. i have never heard of it **Person1:** i do not make any spelling mistakes **Person2:** that is great! **Person1:** i am from california. my hair is brown **Person2:** what is your ethnicity? i am asian.

Table 7.2: Example of a dialogue conversation generated by BART model.

hi , how it goes
i love you.
i do not love you. What do you mean?
i am very glad you came to England
that sounds like a good man
i have not heard of you yet.
i have been, i have been. You are welcome, sir

Table 7.3: Experiment of Beam search decoding method.

hi, how is thy name
i am an ass, and thou shalt be
i am sure you have not seen a man till you
i do not know, Wherefore you speak.
i ride you, And i love it
i am very glad you are come
i have a daughter of old

Table 7.4: Experiment of Nucleus sampling decoding method.

large, which can take a very long time. As there is no visible difference in the results, so in other experiments beam search is used due to time efficient.

7.1 Feature-based decoding modifications

Controlling specificity

In the experiment the Normalized Inverse Document Frequency feature is used as a decoding feature. This approach makes generated text more specific, because the less often a word is found in a dataset the more weight is set for the word probability.

Person 1 description: 1. as a child, i won a national spelling bee. 2. i have been published in the new yorker magazine. 3. i am a gourmet cook. 4. i have perfect pitch.
Person 2 description: 1. i am very athletic. 2. i wear contacts. 3. i have brown hair. 4. i love bicycling. 5. i hate carrots.
Person1: (Silence) **Person2:** hello , how are you doing today? **Person1:** hi! i work as a gourmet cook **Person2:** what kind of gourmet cook do you cook? **Person1:** i do not like carrots. i throw them away. **Person2:** what kind of carrots do you discard? **Person1:** really. but, i can sing pitch perfect. **Person2:** what kind of music do you sing pitch perfect? **Person1:** i also cook, and i ride my bike to work. **Person2:** that sounds fun! i ride my bike to work as well. **Person1:** great! i had won an award for spelling bee. **Person2:** awesome! what kind of spelling bee did you win?

Table 7.5: Generated conversation by BART model with specifity, weight = 5.

Example results of using BART model with NIDF is shown in the Table 7.5. NIDF is multiplied by hand tuned weight to increase the probability of words. The conversation generated by BART model with NIDF uses such specific words as “discard”, “award” and **64.3%** of people consider that the generated dialogue is more specific than real one.

Length control

For this experiment 2 lengths of a sequence were chosen: 5 and 20 words.

hi, how are you tonight?	hi, how are you?
i am a chef. i love to cook.	i am a chef.
i do not like carrots either.....	i do not like carrots
i am very athletic. i ride my bike to work every morning	i am very athletic
i am from the country of birth	i am from texas

Table 7.6: Experiment of generated sequences with setted length.

Generated sequences with this feature are shown in the Table 7.6. In some cases, when generating long sentences, many punctuation marks were added, as in the sentence "i do not like carrots either.....". **71.4%** people consider, that dialogues generated with a lot of words in a sentence are more interesting.

Stop word blocking

Stop words are used from NLTK[1](suite of libraries and programs for symbolic and statistical NLP). During decoding in each iteration, when selecting the next word in the sequence, it is checked if this word is not contained in stop words.

good morning
oh yeah
oh yeah
oh nice sound
awwwwwwwwwwwwww

Table 7.7: Experiment of a generated dialogue with blocking stop words.

Represented results of the experiment in the Table 7.7 shows that generated sequences are very generic and don't make sense in a context. For this experiment, there was no human evaluation.

7.2 Weighted decoding of combination BART and GPT-2 models

Results of this experiment are generated dialogues from combination of BART and GPT-2 models for stylization, the weights for other stylistic language models are set to 0.

Example of a generated dialogue with humor is shown in the Table 7.8. The weight for encoder-decoder model is set to 0.3, the weight for stylized language model is set to 0.7. These hyperparameters were manual tunned. Compared to the generated dialogue using only the BART model (Table 7.2), this conversation is not very different, although, for example, the phrase: "i am a chef for carrots" may seem rather funny, because the opponent does not like carrots and **57.14%** people consider that this dialogue is comical.

Generated dialogue with poetic style is represented in the Table 7.9. Hyperparameters for weighted decoding are set to 0.2 for BART and 0.8 for GPT-2 model. In the example a poetic appeal to a person ("My Lord") is used, which means a manifestation of style. **64.3%** people consider that this dialogue is more poetic than the real one.

In the Table 7.10 GPT-2 is trained on SST dataset. As a result generated sentences are short. It could be because sentences in the courpus are ranging from 10 to 20 words (a histogram of the SST dataset is in the Figure 5.3). The negative and positive moods are not expressed in the example. Generated dialogues were compared with the real conversation and in case of positive sentiment only **21.4%** people consider that generated dialogue is positive, in case of negative sentiment – **50%** people think that generated conversation is more negative than the real one.

General conclusion from this experiment is that the more weight was set on generating the style, the more the context began to get lost. Generated sentences were often unrelated

[1] https://www.nltk.org/

Person 1 description: 1. as a child, i won a national spelling bee. 2. i have been published in the new yorker magazine. 3. i am a gourmet cook. 4. i have perfect pitch.
Person 2 description: 1. i am very athletic. 2. i wear contacts. 3. i have brown hair. 4. i love bicycling. 5. i hate carrots.
Person 1: hey! how are you today? **Person 2:** hi! i work as a gourmet cook **Person 1:** i was just about to go to a gourmet cook shop. **Person 2:** i do not like carrots. i throw them away. **Person 1:** i am a chef for carrots. **Person 2:** really. but, i can sing pitch perfect. **Person 1:** i love the chorus. **Person 2:** i also cook, and i ride my bike to work. **Person 1:** great! i had won an award for spelling bee. **Person 2:** that is awesome. i am sure!

Table 7.8: Experiment of weighted decoding of BART with GPT-2 trained on dataset with jokes.

Person 1 description: 1. as a child, i won a national spelling bee. 2. i have been published in the new yorker magazine. 3. i am a gourmet cook. 4. i have perfect pitch.
Person 2 description: 1. i am very athletic. 2. i wear contacts. 3. i have brown hair. 4. i love bicycling. 5. i hate carrots.
Person1: (Silence) **Person2:** hi there, how shall we get on? **Person1:** i love carrots. i know how you feel. **Person2:** i do not know what to do, My lord. **Person1:** i am not a great singer. **Person2:** i ride to work **Person1:** that is the same. **Person2:** i am not heard of that. **Person1:** i am not very good writer, but I am very good poet. **Person2:** i am not a spelling student. But i have a question for you. **Person1:** i am not a cook, I am a gourmet cook **Person2:** i am not a fool. But I am a man of color

Table 7.9: Experiment of weighted decoding of BART and GPT-2 trained on Shakespeare dataset.

Generated conversation with negative SST stylistic dataset.	**Generated conversation with positive SST stylistic dataset.**
Person 1 description: 1. i am very athletic. 2. i wear contacts. 3. i have brown hair. 4. i love bicycling. 5. i hate carrots.	
Person 2 description: 1. as a child, i won a national spelling bee. 2. i have been published in the new yorker magazine. 3. i am a gourmet cook. 4. i have perfect pitch.	
Person1: hi there?	**Person1:** hi how are you
Person2: i work.	**Person2:** i love cooking
Person1: i do not eat	**Person1:** i am not
Person2: i am a singer	**Person2:** i have you heard
Person1: that is awesome	**Person1:** that sounds fun.
Person2: that is cool?	**Person2:** that is pretty?
Person1: i have not.	**Person1:** i do you know.
Person2: i have been.	**Person2:** i am very.
Person1: i do not know	**Person1:** i am sure.
Person2: i am very good	**Person2:** i am a professional
Person1: i am white	**Person1:** i am black
Person2: i am asian	**Person2:** i am asian

Table 7.10: Experiment of weighted decoding of BART and GPT-2 trained on SST stylistic dataset.

in context. Therefore, the following experiment was conducted with the alternate use of models.

7.3 Switching BART and GPT-2 models

In this experiment, alternating the generation of words by the encoder-decoder and stylized language models. n words in a sequence are generated over BART and n words are generated over GPT-2 model, where n is a hyperparameter. Sequence generation stops when the end mark of the sequence is generated or the length of the sequence is equal to the maximum length.

Example of the experiment is represented in the Table 7.11. When generating sequences, there is still a problem of context, but already the model sometimes generates answers to the opponent's input ("-i love carrots but I hate carrots too. -i don't like it either."). **71.4%** people consider that this dialogue is more comic than the real one.

Person 1 description: 1. i am very athletic. 2. i wear contacts. 3. i have brown hair. 4. i love bicycling. 5. i hate carrots.
Person 2 description: 1. as a child, i won a national spelling bee. 2. i have been published in the new yorker magazine. 3. i am a gourmet cook. 4. i have perfect pitch.
Person1: hi there. how much do you charge? **Person2:** i love carrots but I hate carrots too. **Person1:** i don't like it either. **Person2:** i am a singer. **Person1:** i ride the bike to work. The weather is beautiful **Person2:** i am not sure. i have never seen that. What is the spelling of the word **Person1:** i don't know. **Person2:** i am a gourmet chef. **Person1:** i am not a spelling bee. **Person2:** i am not a cook. What is your favorite food? I would love to know. **Person1:** i am a white man **Person2:** i am a asian

Table 7.11: Experiment of switching BART and GPT-2 models trained on jokes stylistic dataset.

The sentiment and poetic style are not expressed in the examples shown in the Tables 7.12, 7.13. For the conversation generated by the model fine-tuned on the Shakespeare dataset only **37.5%** of people consider this dialogue poetic. **33.3%** is a human evaluation for negative sentiment and **23.8%** is for positive sentiment.

Person 1 description: 1. i am very athletic. 2. i wear contacts. 3. i have brown hair. 4. i love bicycling. 5. i hate carrots.
Person 2 description: 1. as a child, i won a national spelling bee. 2. i have been published in the new yorker magazine. 3. i am a gourmet cook. 4. i have perfect pitch.
Person1: hi , i am jane **Person2:** i love to cook. i am a gourmet. **Person1:** i do not like carrots. i am allergic **Person2:** i am a very athletic young man, and you? **Person1:** i ride my bike, but I do fear of heights. **Person2:** that is great **Person1:** i am a goner **Person2:** i have been published. **Person1:** i have done a lot. i am very good **Person2:** i am a chef . is a gourmet too **Person1:** i am not sure of my ethnicity **Person2:** i am not sure of what that is.

Table 7.12: Experiment of switching BART and GPT-2 models trained on Shakespeare dataset.

Generated conversation with negative SST stylistic dataset.	**Generated conversation with positive SST stylistic dataset.**
Person 1 description: 1. i am very athletic. 2. i wear contacts. 3. i have brown hair. 4. i love bicycling. 5. i hate carrots.	
Person 2 description: 1. as a child, i won a national spelling bee. 2. i have been published in the new yorker magazine. 3. i am a gourmet cook. 4. i have perfect pitch.	
Person1: hi there, how much do you care about?	**Person1:** hi, who wrote that poem?
Person2: i love to cook, but I do not think i have ever been more captivating	**Person2:** i love to cook, and i am ...
Person1: i don't like it when the carrots come out	**Person1:** i don't like, i n't like carrots, i don't like vinegar
Person2: i have a lot of friends	**Person2:** i am not a great writer.
Person1: i ride the bike of course	**Person1:** i ride his bike to and from the office.
Person2: i have to be a bit more of a nerd	**Person2:** that is, i. e. i am not allowed to use it.
Person1: i have a lot of fun	**Person1:** i have a lot of fun.
Person2: i have a hard time believing that a movie is that good	**Person2:** i have a lot of fun. it is one of my all-time favourite books.
Person1: i amiable and kind of sweet	**Person1:** i am not a spelling mistake
Person2: i am not sure	**Person2:** i am not a cook, but rather a chef
Person1: i am not sure I can believe it	**Person1:** i am not a white man, but i am a black man
Person2: i do n't like it	**Person2:** i am not sure why

Table 7.13: Experiment of switching BART and GPT-2 models trained on SST stylistic dataset.

7.4 Evaluating the evaluation

Fisher's exact test is carried out to check human evaluation for statistically significant biases in it, with the help of which the independence of the dialogue from the order of its display is checked. Two questionnaires are created for the test. In the first one a dialogue with poetic style is displayed as first and real conversation is shown as a second dialogue after that the participant chooses a more poetic dialogue. The second questionnaire is almost the same, only the order in which dialogs are displayed is changed.

The Table 7.14 represents results of the responses of the questionnaire. If the generated dialogue was shown first, then out of 7 people only 4 people chose it as a more stylish dialogue. If the real dialogue was shown first, then out of 8 people, only 3 people chose the generated conversation as a more stylish one.

To use Fisher's exact test, it is necessary that no more than 80% of the expected values are more than 5. For the experiment this condition is satisfied (Table 7.15).

$$p = \frac{\binom{7}{4}\binom{8}{3}}{\binom{15}{7}} = \frac{\binom{7}{3}\binom{8}{5}}{\binom{15}{8}} = 0.305 \tag{7.1}$$

	Generated dialogue	Real dialogue	Total
Generated dialogue first	4	3	7
Real dialogue first	3	5	8
Total	7	8	15

Table 7.14: Results from questionaires.

	Generated dialogue	Real dialogue
Generated dialogue first	3.267	3.733
Real dialogue first	3.733	4.267

Table 7.15: Expected values.

In the Equation 7.1 the probability for null hypothesis is represented. In this case it is not necessary to calculate extreme deviations because the probability is already more than 5%, so null hypothesis is confirmed, that there is no bias between the arrangement of dialogs and the choice of users.

Chapter 8

Conclusion

The goal was to create a model that generates a stylistic response to the sequence of an opponent in the dialogue, in order to give the impression that the proposals are generated by one person.

In this thesis I have focused on several aspects of style manifestation: specificity, sentiment, humor, poetry. Balance between specificity and genericness is one of the important factors that helps make the generated text more diverse. The mood of the interlocutor is also a key factor in human-human conversation. Each person has emotions, and a neutral dialogue gives a human the feeling that he is talking to a bot. Poetry and humor are a manifestation of high intellectual abilities that are inherent only to man.

For these purposes, an architecture of the model was created that combines several language models. For this model transformer-based and LSTM-based architectures were used. The experiments were conducted not only with the created model, but also with different feature-based modifications during the test time for controlling specificity. Generated sequences by decoding modifications with sentences lengths and word rareness were evaluated as more specific. Experiments with weighted decoding and combinations of models have shown that the more weight is set for a model generating stylistic text and the less weight is set for a model generating text depending on input, the less a dialogue makes sense. This is due to the fact that stylistic datasets do not contain dialogs and stylistic language models generate text regardless of input. In the case of alternate use of models, the text becomes more reasonable, but the style is no longer expressed so clearly.

A potential continuation of this work could collect datasets that contain stylistic dialogues rather than expressions independent of each other. If these datasets are created, it will be possible to use stylistic language models with seq2seq architecture.

Bibliography

[1] Alder, H. *Handbook of NLP: A manual for professional communicators.* Routledge, 2017.

[2] Artstein, R., Gandhe, S., Gerten, J., Leuski, A. and Traum, D. Semi-formal evaluation of conversational characters. In: *Languages: From formal to natural.* Springer, 2009, p. 22–35.

[3] Banerjee, S. and Lavie, A. 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.* 2005, p. 65–72.

[4] Bengio, Y., Ducharme, R., Vincent, P. and Jauvin, C. A neural probabilistic language model. *Journal of machine learning research.* 2003, vol. 3, Feb, p. 1137–1155.

[5] Bengio, Y., Simard, P. and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks.* IEEE. 1994, vol. 5, no. 2, p. 157–166.

[6] Berg, M. M. *Modelling of natural dialogues in the context of speech-based information and control systems.* Dissertation.

[7] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F. et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *ArXiv preprint arXiv:1406.1078.* 2014.

[8] Fisher, R. A. On the interpretation of χ 2 from contingency tables, and the calculation of P. *Journal of the Royal Statistical Society.* JSTOR. 1922, vol. 85, no. 1, p. 87–94.

[9] Flesch, R. F. *How to write plain English: A book for lawyers and consumers.* Harpercollins, 1979.

[10] Gao, X., Zhang, Y., Lee, S., Galley, M., Brockett, C. et al. Structuring latent spaces for stylized response generation. *ArXiv preprint arXiv:1909.05361.* 2019.

[11] Ghazvininejad, M., Shi, X., Priyadarshi, J. and Knight, K. Hafez: an interactive poetry generation system. In: *Proceedings of ACL 2017, System Demonstrations.* 2017, p. 43–48.

[12] Giménez, J. and Màrquez, L. A smorgasbord of features for automatic MT evaluation. In: *Proceedings of the Third Workshop on Statistical Machine Translation*. 2008, p. 195–198.

[13] Gkatzia, D. and Mahamood, S. A snapshot of NLG evaluation practices 2005-2014. In: *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*. 2015, p. 57–60.

[14] He, K., Zhang, X., Ren, S. and Sun, J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 770–778.

[15] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*. MIT Press. 1997, vol. 9, no. 8, p. 1735–1780.

[16] Holtzman, A., Buys, J., Forbes, M. and Choi, Y. The curious case of neural text degeneration. *ArXiv preprint arXiv:1904.09751*. 2019.

[17] Jhamtani, H., Gangal, V., Hovy, E. and Nyberg, E. Shakespearizing modern language using copy-enriched sequence-to-sequence models. *ArXiv preprint arXiv:1707.01161*. 2017.

[18] John, V., Mou, L., Bahuleyan, H. and Vechtomova, O. Disentangled representation learning for text style transfer. *ArXiv preprint arXiv:1808.04339*. 2018, p. 1301–1310.

[19] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *ArXiv preprint arXiv:1412.6980*. 2014.

[20] Kohavi, R. and Longbotham, R. Online Controlled Experiments and A/B Testing. *Encyclopedia of machine learning and data mining*. 2017, vol. 7, no. 8, p. 922–929.

[21] Lee, C. van der, Gatt, A., Miltenburg, E. van, Wubben, S. and Krahmer, E. Best practices for the human evaluation of automatically generated text. In: *Proceedings of the 12th International Conference on Natural Language Generation*. 2019, p. 355–368.

[22] Lemaréchal, C. Cauchy and the gradient method. *Doc Math Extra*. 2012, vol. 251, p. 254.

[23] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A. et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *ArXiv preprint arXiv:1910.13461*. 2019.

[24] Li, J., Jia, R., He, H. and Liang, P. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *ArXiv preprint arXiv:1804.06437*. 2018.

[25] Lin, C.-Y. ROUGE: A Packagefor Automatic Evaluation of Summaries. In: *ProceedingsofWorkshop on Text Summarization Branches Out, Post2Conference Workshop of ACL*. 2004.

[26] Liu, C.-W., Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L. et al. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *ArXiv preprint arXiv:1603.08023*. 2016.

[27] Luong, M.-T., Pham, H. and Manning, C. D. Effective approaches to attention-based neural machine translation. *ArXiv preprint arXiv:1508.04025*. 2015.

[28] Manishina, E. *Data-driven natural language generation using statistical machine translation and discriminative learning*. Dissertation.

[29] Napoles, C., Sakaguchi, K. and Tetreault, J. There's No Comparison: Reference-less Evaluation Metrics in Grammatical Error Correction. *ArXiv preprint arXiv:1610.02124*. 2016.

[30] Novikova, J., Dušek, O., Curry, A. C. and Rieser, V. Why we need new evaluation metrics for NLG. *ArXiv preprint arXiv:1707.06875*. 2017.

[31] Oh, A. H. and Rudnicky, A. I. Stochastic natural language generation for spoken dialog systems. *Computer Speech & Language*. Elsevier. 2002, vol. 16, 3-4, p. 387–407.

[32] Pang, B. and Lee, L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: Association for Computational Linguistics. *Proceedings of the 43rd annual meeting on association for computational linguistics*. 2005, p. 115–124.

[33] Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J. BLEU: a method for automatic evaluation of machine translation. In: Association for Computational Linguistics. *Proceedings of the 40th annual meeting on association for computational linguistics*. 2002, p. 311–318.

[34] Peng, N., Ghazvininejad, M., May, J. and Knight, K. Towards controllable story generation. In: *Proceedings of the First Workshop on Storytelling*. 2018, p. 43–49.

[35] Pennington, J., Socher, R. and Manning, C. D. Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, p. 1532–1543.

[36] Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*. 2018.

[37] Ritter, A., Cherry, C. and Dolan, W. B. Data-driven response generation in social media. In: Association for Computational Linguistics. *Proceedings of the conference on empirical methods in natural language processing*. 2011, p. 583–593.

[38] Rudnicky, A. and Oh, A. H. Dialog annotation for stochastic generation. Carnegie Mellon University. 2002.

[39] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. Learning representations by back-propagating errors. *Nature*. Nature Publishing Group. 1986, vol. 323, no. 6088, p. 533–536.

[40] Salovey, P. and Mayer, J. D. Emotional intelligence. *Imagination, cognition and personality*. Sage Publications Sage CA: Los Angeles, CA. 1990, vol. 9, no. 3, p. 185–211.

[41] See, A., Roller, S., Kiela, D. and Weston, J. What makes a good conversation? how controllable attributes affect human judgments. *ArXiv preprint arXiv:1902.08654*. 2019.

[42] Sennrich, R., Haddow, B. and Birch, A. Controlling politeness in neural machine translation via side constraints. In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016, p. 35–40.

[43] Shang, L., Lu, Z. and Li, H. Neural responding machine for short-text conversation. *ArXiv preprint arXiv:1503.02364*. 2015.

[44] Shen, T., Lei, T., Barzilay, R. and Jaakkola, T. Style transfer from non-parallel text by cross-alignment. In: *Advances in neural information processing systems*. 2017, p. 6830–6841.

[45] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*. JMLR. org. 2014, vol. 15, no. 1, p. 1929–1958.

[46] Stent, A., Marge, M. and Singhai, M. Evaluating evaluation methods for generation in the presence of variation. In: Springer. *International conference on intelligent text processing and computational linguistics*. 2005, p. 341–351.

[47] Sutskever, I., Vinyals, O. and Le, Q. V. Sequence to sequence learning with neural networks. In: *Advances in neural information processing systems*. 2014, p. 3104–3112.

[48] Tikhonov, A. and Yamshchikov, I. P. Guess who? Multilingual approach for the automated generation of author-stylized poetry. In: IEEE. *2018 IEEE Spoken Language Technology Workshop (SLT)*. 2018, p. 787–794.

[49] Vaidyam, A. N., Wisniewski, H., Halamka, J. D., Kashavan, M. S. and Torous, J. B. Chatbots and conversational agents in mental health: a review of the psychiatric landscape. *The Canadian Journal of Psychiatry*. SAGE Publications Sage CA: Los Angeles, CA. 2019, vol. 64, no. 7, p. 456–464.

[50] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L. et al. Attention is all you need. In: *Advances in neural information processing systems*. 2017, p. 5998–6008.

[51] Wen, T.-H., Vandyke, D., Mrksic, N., Gasic, M., Rojas Barahona, L. M. et al. A network-based end-to-end trainable task-oriented dialogue system. *ArXiv preprint arXiv:1604.04562*. 2016.

[52] Wheatley, J. The computer as poet. *Queen's Quarterly*. Quarterly Committee of Queen's University. 1965, vol. 72, no. 1, p. 105.

[53] Zhang, S., Dinan, E., Urbanek, J., Szlam, A., Kiela, D. et al. Personalizing Dialogue Agents: I have a dog, do you have pets too? *ArXiv preprint arXiv:1801.07243*. 2018.

[54] Zhou, H., Huang, M., Zhang, T., Zhu, X. and Liu, B. Emotional chatting machine: Emotional conversation generation with internal and external memory. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.