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 **AUTOR PRÁCE**	**KSENIA BOLSHAKOVA**
SUPERVISOR **VEDOUCÍ PRÁCE**	**Ing. MARTIN FAJČÍK**

BRNO 2020

Abstract

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

Abstrakt

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

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

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
April 17, 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 (i.e. conversational agent) 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, which has a significant impact on system quality, because NLG goal is to imitate human behaviour. The conversational agent can be classified into task-oriented (i.e. goal-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 have incorporated goal-oriented dialogue systems, such as Yandex's Alisa, Apple's Siri, Microsoft's Cortana, Amazon Alexa, and Google Assistant. Goal-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 conversational agent often have an entertainment value, such as Cleverbot, Microsoft's XiaoIce system etc. Chatbots have also been used for testing theories of psychological counseling.

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 researches work on open-ended (i.e. there is a huge range of appropriate outputs given the input) chitchat dialogues to explore this aspect of intelligence, because in goal-oriented dialogue systems there is a relatively narrow range of correct outputs given the input. Creating a non-task-oriented agent is a challenge for researches, because there are a lot of topics of conversations as well as user reactions and responses to them. Such bots are not able to generate meaningful conversations. Their replies are often too generic, because non-specific responses sound quite natural (e.g. "Ok, I see", "I don't know"). There are still a lot of problems in the Natural Language Generation, such as response-relatedness, semantic errors, repetition etc., which will be described in more detail in the chapture 2.

According to [28] 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 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 NLG model, which will be able to generate text in different styles. //TODO

Chapter 2

NLG problems in dialogue systems

In a book [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 corpora is a collection of texts [21].

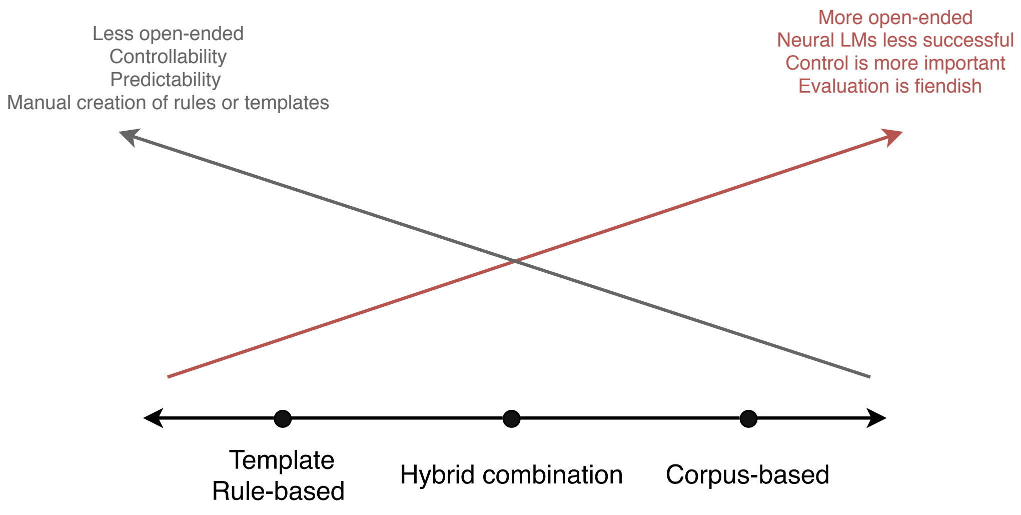

Figure 2.1: Spectrum of the NLG tasks.

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 [19]. 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*}?
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 output produced by this approach is likely to be grammatically correct and not contain unexpected generation errors. 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). These systems are not able to handle unknown inputs. Templates often sound unnatural due to their generic structures. Template-based systems cannot make variation in output, it is just concatenation of strings. This approach is not flexible, because it has limits to use templates in other domains. 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 system dominates the NLG community, special 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 it [27].

One of the first approaches in corpus-based methods is **Dynamic Sentence Generation**, which dynamically creates sentences from representations of the meaning to be conveyed by the sentence and/or its desired linguistic structure. It allows do not write code for every boundary case and includes aggregation, reference, ordering and connectives to optimise sentences.

Next level of corpus-based approaches is **Dynamic Document Creation**, what can produce relevant and well-structured document.

Advantages of corpus-based approach

Corpus-based models have ability to generate more proper responses that could have never appeared in the corpus; it is possible to mimick the language of a real domain expert and use this models for open-domain dialogue systems; 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 and on a variety of topics to get a sensible output. Even if you have the corpus, process of text generation is not fully controlled and the output can be incorrect or does not make a sence. This approach still has a lot of problems, what will be described in more detail in the section 2.4.

2.3 Language Models

In corpus-based system natural language generation uses **Language Models(LMs)** to generate sequences of texts. LM is a probabilistic model which learns to predict 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. The language model provides a context for distinguishing words and phrases that sound the same. For example the phrases "but her" and "butter" sound the same, but mean different things.

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

The **Chain rule** (equation 2.2) is used to calculate the joint probability of a sentence by using the conditional probability (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.

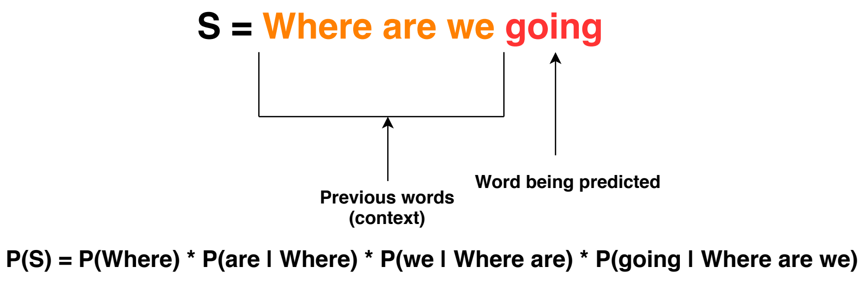
S = Where are we going
Previous words (context)
Word being predicted
P(S) = P(Where) * P(are I Where) * P(we I Where are) * P(going I Where are we)

Figure 2.2: Example of a word prediction 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

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 the occurrence of an N-gram in a sequence of words can be useful, because it can help 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 [3] another 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 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 Dialogue systems

The ability to communicate with machines in a natural language is a long-standing dream of 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 [5] 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.

1	More phones have games on them than this one.
2	Why a mouse when it spins?

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.

1	They IS going to school.
2	It depends AT you.

Table 2.4: A problem of syntactic correctness.

As noticed in [34], 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 not only syntactic correctness of output but also if output is appropriate or felicitous in a given context. A good generator usually relies on several factors:

-Yes, I'm studying law at the moment. - Good. - I like playing the piano. - Good.

Table 2.5: A problem of repetition makes conversation boring.

-Do you go get coffee often? -I am a musician.

Table 2.6: 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.

- **adequacy** (a sentence that is ambiguous or not contains communicates meaning in the input, is **not** adequate (an example in the Table 2.3))
- **syntactic correctness** (an example in the Table 2.4)
- **repetition** (self-repetition across utterances and with utterances, repeating the conversational partner (an example in the Table 2.5))
- **response-relatedness** (efficacy in context (an example in the Table 2.6))
- **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 2.7: The example of sentences' variation.

An example in the Table 2.7 shows all types of variation. Sometimes this factor can be syntactically incorrect or unclear, what you can see in the forth sentence. In fifth sentnecne 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

Models for Natural Language Generation

NLG evolution from templates 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 utterance. Thus, these models fit well within the sequence-to-sequence (seq2seq) (i.e. encoder-decoder) models with using neural networks (NNs), which are described in more detail in section 3.4, but first a short description what neural networks are, for a better understanding seq2seq model.

3.1 Neural Networks

Artificial Neural Networks are inspired by biological neural networks that constitute animal brains. Artificial neuron (on the 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 either right or left.

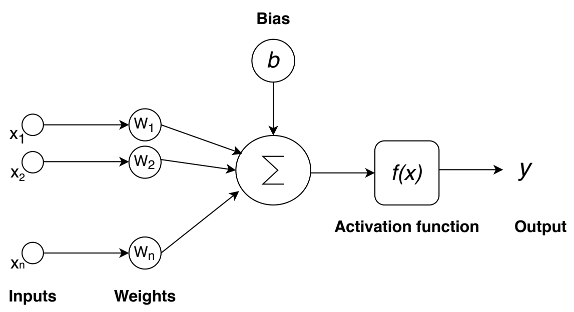

Figure 3.1: Architecture of an artificial neuron.

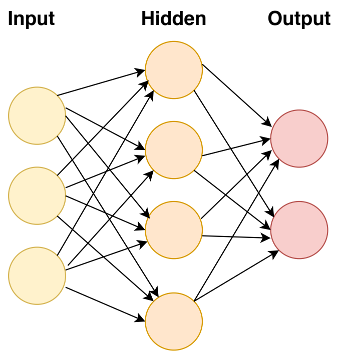

Figure 3.2: The fully-connected neural network with 1 hidden layer.

Neural networks are acyclic directed graphs of neurons. Neurons' outputs can be connected to inputs of other neurons and the calculation is propagated through the network. In 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.

3.2 Recurrent neural network (RNN)

Recurrent neural networks are a special type of a neural network used in natural language processing(NLP) as they allow temporal depedencies 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.

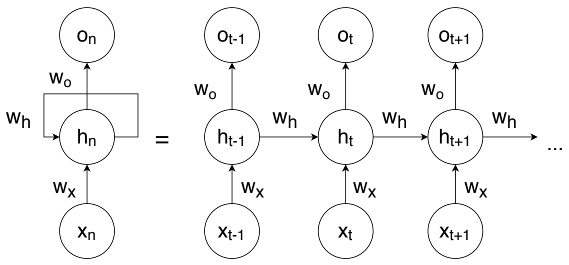

Figure 3.3: Architecture of a recurrent neural network, where coefficient **h** is a hidden state, **x** is an input and **o** is an output. Coefficient **w** is a weight, what is transformed to produce a sensible output.

The architecture of RNN is illustrated in the Figure 3.3. A hidden state h is realized as 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 goal-oriented dialogue system [39] and a training model with semantic aggregation [37].

Nowadays traditional RNN networks almost are not used in NLG, because they have problems with vanishing and exploding gradients. As introduced in [4] the exploding problem refers to the large increase in the norm of the gradient 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 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 [10].

3.3 Long short-term memory (LSTM)

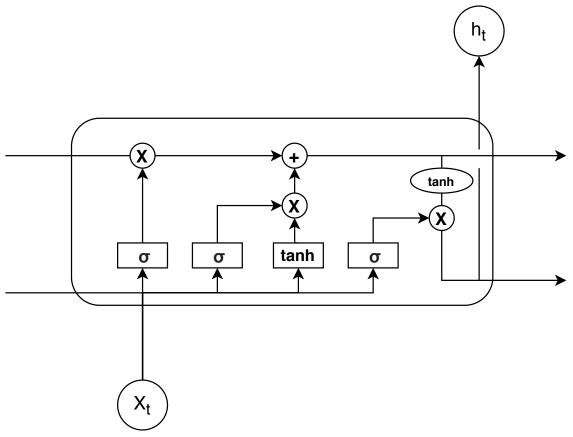

Figure 3.4: A cell in an LSTM network.

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 module has a different structure. 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. 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).

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

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 have a problem with vanishing gradient, but still the capacity of the LSTM memory is limited, because of inherently complex sequential words' paths from the previous unit to the current unit. The same complexity results in high computational requirements that make LSTM difficult to train.

3.4 Sequence-to-sequence model (seq2seq)

Seq2seq models were introduced by Google in 2014 [35]. This model uses an encoder-decoder architecture (the 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 2 neural networks are connected into one model to maximize the learning effect. Seq2seq model is very effective to solve NLP problems, because input and output sequences can have different lengths and recurrent neural networks can work with context. This model is often used in machine translation, text summarization, dialogue systems etc.

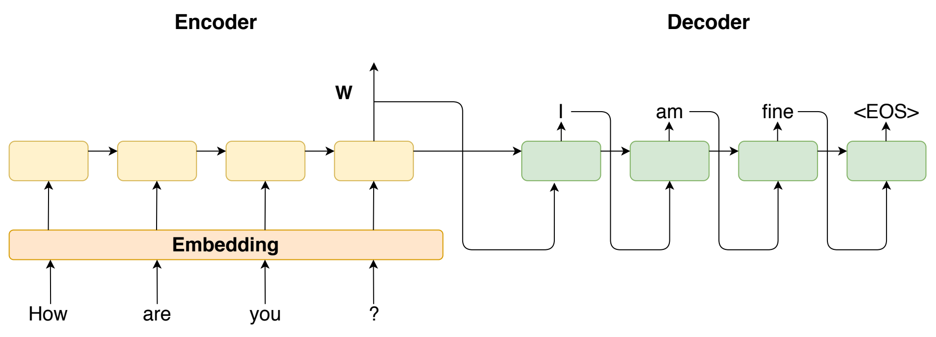

Figure 3.5: Architecture of sequence-to-sequence model.

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 word and the context of the word. 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. Many optimizations have led to other seq2seq components, such as attention, beam search, bucketing.

3.5 Attention

A neural attention mechanism is based on 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.

In [38] 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 is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

The attention mechanism provides the decoder with information from each hidden state of the encoder and it gives a model the ability to selectively focus on useful parts of the input sequence and learn the alignment between them (example in the Figure 3.6).

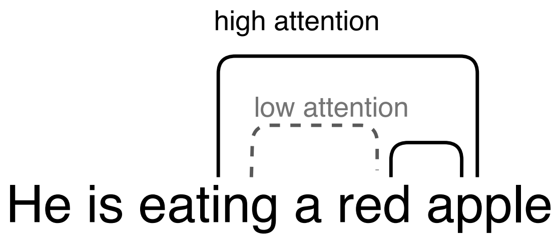

Figure 3.6: When we see "eating", we expect to encounter a food word very soon. The color term describes the food, but probably not so much with "eating" directly.

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. Formally, in the 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.

3.6 Transformer

Information in this section is taken from [38].

Transformer introduces an architecture that is based on self-attention mechanism and does not use any recurrent networks. In each step this model applies self-attention mechanism which directly models relationships between all words in a sequence, regardless of their respective position. Transformers do not require that the sentence be processed in

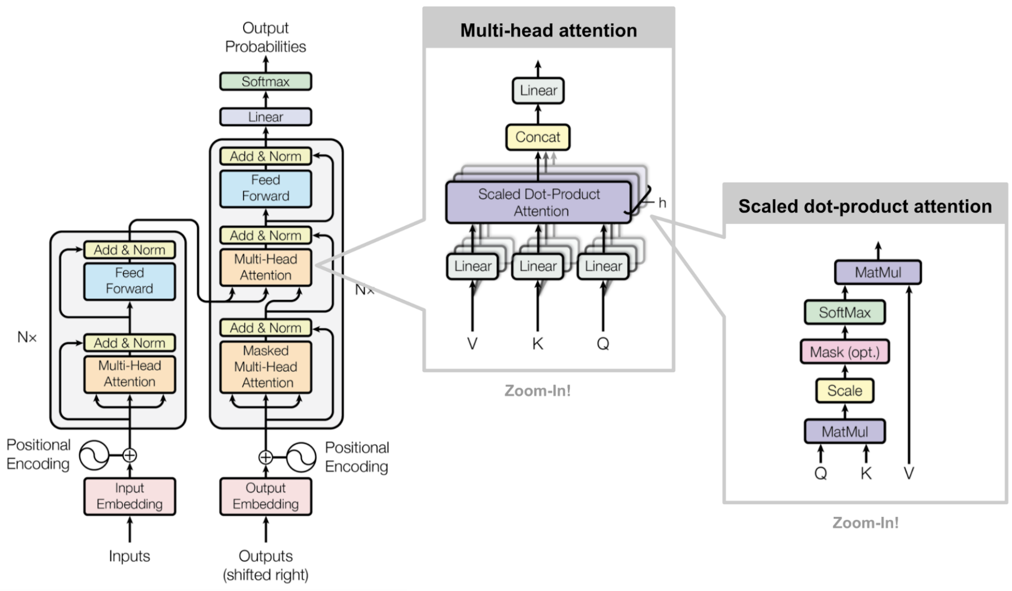

Figure 3.7: The architecture of the Transformer.[1]

order, that allows process parallelization during training, unlike RNN. Due to this feature, it has enabled training on much more data.

The architecture of the Transformer is illustrated in the Figure 3.7. The Transformer consists of a stack of encoders (on the left) for processing inputs of any length and another set of decoders (on the right) to output the generated sentences. N_x in the Figure means that modules of encoder and decoder can be stacked on top of each other multiple times. Modules consist of multi-head attention and feed forward layers. The inputs and output are first embedded into an n-dimensional space. Words' positions are added to the embedded representation, n-dimensional vector, of each word.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3.9}$$

$$MultiHead(Q,K,V) = Concat(head_1, head_2, ..., head_h)W^O \tag{3.10}$$

An equation 3.9 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 matrices.

An equation 3.10 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.

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

3.7 GPT

Information in this section is taken from [25]. GPT is a transformer-based (Section 3.6) 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 (There are only input variables and no correspoding output variables in unsupervised dataset.), then the model is fine-tuned on much smaller supervised dataset (There are input and output variable in supervised dataset. The model learns the mapping function from the input to the output.) to help it solve specific tasks.

Unsupervised pre-training

In unsupervised pre-training a standard language modeling is used and the aim is to maximize the likelihood (Equation 3.11, 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).

$$L_1(U) = \sum_i \log P(u_i | u_{i-k}, ..., u_{i-1}; \Theta) \tag{3.11}$$

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 3.12, 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).

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_1 &= transformer\_block(h_{l-1}) \forall l \in [1, n] \\ P(u) &= softmax(h_n, W_e^T) \end{aligned} \tag{3.12}$$

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 3.13).

$$P(y|x^1, ..., x^m) = softmax(h_l^m, W_y) \tag{3.13}$$

$$L_2(C) = \sum_{(x,y)} \log P(y|x^1, ..., x^m) \tag{3.14}$$

Authors found that including language modeling 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 3.15).

$$L_3(C) = L_2(C) + \lambda * L_1(C) \tag{3.15}$$

3.8 BART

BART is a denoising autoencoder for pretraining sequence-to-sequence models, that maps a corrupted document to the original document it was derived from. The model uses a standard Transformer-based neural machine translation architecture, with a bidirectional encoder and left-to-right decoder.

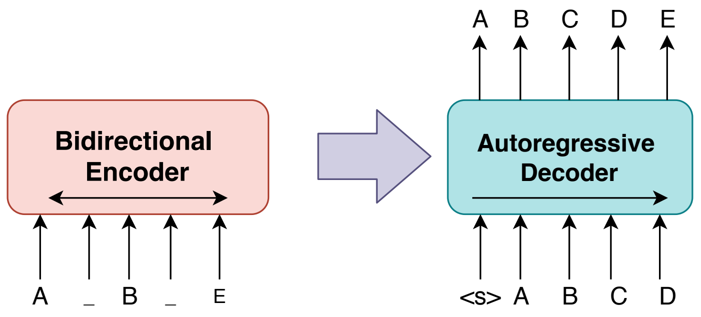

Figure 3.8: BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder[15].

Bidirectional encoder consists of N transformer encoder blocks stacked on top of each other (Section 3.6), the final block is the output. 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. Bidirectional encoder uses 2 training strategies: **Masked LM (MLM)** and **Next Sentence Prediction (NSP)**.

In **MLM** 15% 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. Pretraining on this task allows the model to learn general features of the language.

Next Sentence Prediction. The bidirectional encoder in the training process receives pairs of sequences as input and the task of the model is to say whether the second sentence is a subsequent of the first sentence in the original document. During training, part of the inputs are a pair in which the second sentence is a subsequent in the original document and in the other part a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence. Pre-training on this task allows the model to learn relationships between sentences.

$$f(x) = x^+ = max(0, x) \tag{3.16}$$

Equation 3.16: Rectified Linear Unit (ReLU)

$$GELU(x) = xP(X \leq x) = x\Phi(x) \tag{3.17}$$

Equation 3.17: Gaussian Error Linear Unit (GELU)

$$\begin{aligned} &GELU(x) = 0.5x(1 + tanh[\sqrt{2/\pi}(x + 0.044715x^3)]) \\ &GELU(x) = x\sigma(1.702x) \end{aligned} \tag{3.18}$$

Equation 3.18: Approximations of GELU

Autoregressive decoder is GPT model (Section 3.7). Authors modified ReLU activation functions (Equation 3.16) to GELU (Equation 3.17) and initialized parameters from $\mathcal{N}(0, 0.2)$. Autoregressive decoder can be directly fine tuned for sequence generation tasks.

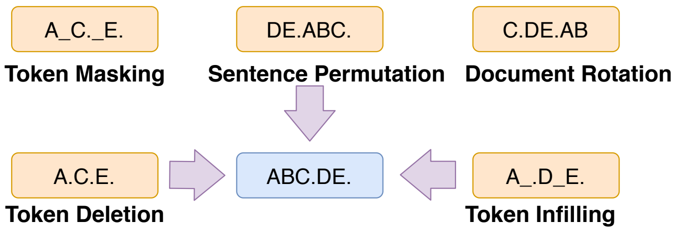

Figure 3.9: Transformations for noising the input. These transformations can be composed.

BART allows to apply any type of document corruption, such as token masking, sentence permutation, document rotation, token deletion, text infilling (Figure 3.9).

Chapter 4

Related works

This chapter presents an overview of the most popular NLG models for building open-ended dialogue systems and for resolving problems described in the section 2.4. Section 4.1 presents approaches for resolving standard NLG problems. Section 4.2 presents solutions for generating stylized text, which helps to create the feeling that you are talking to a person.

4.1 Approaches for dialogue systems problems

Modeling conversations with generative probabilistic models was first proposed in [26]. They present a data-driven approach to generating responses to Twitter status posts, based on phrase-based Statistical Machine Translation. In [31] 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.

Text generation methods

In the paper [29] solutions to common NLG problems in dialogue systems 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 utterance. In this work, authors use **Conditional Training (CT)** [23] and **Weighted Decoding (WD)** [7].

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 (the Figure 4.1). For example, to get very generic or very specific response, z can be set to LOW or HIGH. 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 attributes without sufficient training data. CT model learns only a very weak connection between z and the semantic relatedness of the output.

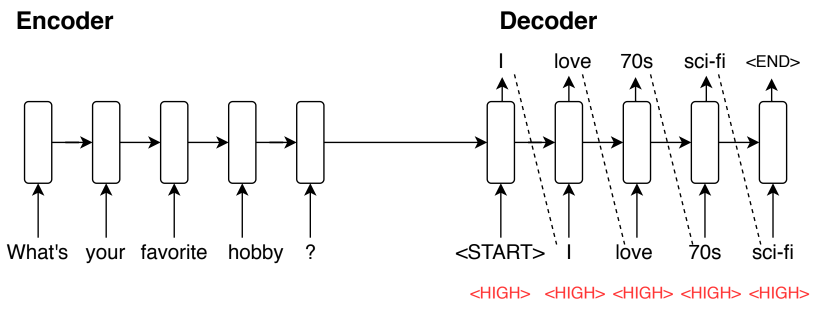

Figure 4.1: Example of Condition Training model.

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 particial 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 calculated by the RNN. $score(y_{<t}; x)$ 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 condition 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).

Question-asking problem is resolved in CT by setting the variable z to 1 of 11 possible values: $\{0, 1, ..., 10\}$, where $z = i$ means that the model should produce, on average, utterances containing "?" with probability $\frac{i}{10}$. In weighted decoding the binary decoding feature is used, which is equal to 1 if a word w is in a pre-defined list of words: "how, what, when, where, which, who, whom, whose, why, ?", 0 otherwise.

N-gram based decoding features were defined to control repetition with WD. External (self-repetition across utterances), internal (self-repetition within utterances) and partner (repeating the conversational partner) repetition fetures identify repeating bigrams. Other features identify repeating content words. Negative weight is applied to these features to reduse repetition.

$$resp\_rel(w; y_{<t}, x) = cos\_sim(word\_emb(w), sent\_emb(l)) \tag{4.3}$$

Response-relatedness feature with weighted decoding is represented in the Equation 4.3, where $word\_emb(w)$ is the GloVe embedding for the word w, $sent\_emb(l)$ is the sentence embedding for the partner's last utterance l (l is part of the context x), cos_sim is the cosine similarity between two. The control response-relatedness in condition training is represented by variable z as $cos\_sim(sent\_emb(y), sent\_emb(l))$, where cos_sim is a cosine similarity between the model's response y and the partner's last utterance l.

Decoding strategies

According to [11] decoding strategies with likelihood maximazing lead to text that is increadibly 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.

Maximization-based decoding methods 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.

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 4.4, 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.

$$p(x = V_l | x_{1:i-1}) = \frac{exp(u_l/t)}{\sum_{l'} exp(u'_l/t)} \tag{4.4}$$

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 4.5.

$$P'(x|x_{1:i-1}) = \begin{cases} P(x|x_{1:i-1})/p' & \text{if } x \in V^{(k)} \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

The research 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 4.6.

$$\sum_{x \in V^{(p)}} P(x|x_{1:i-1}) \geqslant p \tag{4.6}$$

4.2 Approaches for generating stylized conversation

The idea that computers can generate stylized texts appeared half a century ago in [40]. Authors choose a variety of text characteristics as style, such as an expesion of politeness in machine translation [30], transformation from modern English to Shakespearean English [12], sentiment of a text in [32] and [16]. In [6] authors used a structured latent space to generate stylized dialogue responses. Another approach was proposed in [13], where authors applied an adversarial loss to separate style from content. The problem of style transfer, described in these papers, differs from the stylized text generation, because as it is shown in [9] an existing human-written source used to control the reliability of the results can significantly improve the quality of the resulting texts.

ECM

Emotional Chatting Machine (ECM), represented in [42], 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, in the opinion og the authors, emotions are highly subjective.

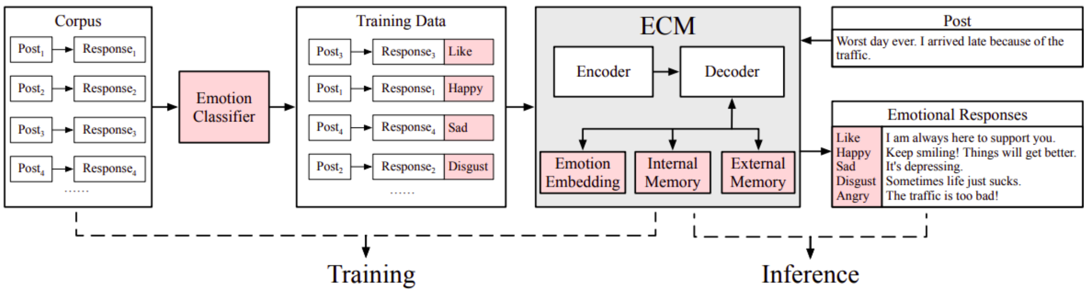

Figure 4.2: Overview of ECM (the grey unit). The pink units are used to model emotion factors in the framework.

In the architecture (the 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 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. In **Emotion Category Embedding** randomly initialize the vector of an emotion category v_e for each category e. During the training the model is learning the vectors of the emotion category. **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. **External memory** is used to model emotion expressions explicitly, the model can choose to generate words from an emotion vocabulary or a generic vocabulary.

ECM need an external decision maker, because this model has to specify an emotion category to be generated.

Generation of stylized texts

In the paper [36] 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. They presented a version of a language model based on a LSTM artificial neural network with extended phonetic and semantic embeddings for stylized poetry generation. The model generates texts resembling the writing style of a particular author.

$$G(C|S) = \begin{cases} (C, \mathbb{R}^m, F) \to \{T_i^G\} \\ \{T_i^G|S\} \sim \{T_i|S\} & \text{w.r.t. D} \end{cases} \tag{4.7}$$

The equation 4.7 represents the stylized model, where $C = \{T_i\}_{i=0}^{M}$ is a corpus of M literary texts written in one natural language. Every text of length l is a sequence $T_i = (w_j)_{j=0}^{l}$ (w_j is a word). Words are drawn from a vocabulary set $V = \{w_j\}_{j=1}^{L}$, L is a vocabulary size. $(C, \mathbb{R}^m, F)$ is all information available to us.

Performance metric D usually tries to minimize $D(\{T_i\}, \{T_i^G\})$, where $\{T_i^G\}$ is a randomized sample of C. S is a subset of continuous and categorical variables out of $(\mathbb{R}^m, F)$ and metric D. Artificial neural networks are used for language modeling to avoid the dimensionality curse by effective mapping $(C, \mathbb{R}^m, F) \to \mathbb{R}^d$ and then train the model like that $G(C) : \mathbb{R}^d \to \{T_i^G\}$.

Advantage of this model is a customization. To control cetain parameters of the model, it is needed to include them intp S. The output $\{T_i^G|S\}$ will resemble original texts $\{T_i|S\}$ that satisfy S conditions. The name of an author of a poetic text is used as a condition S for the model.

Another key feature of the proposed model is a concatenated word representation, where one of the LSTMs works with letters from char-representation of the word and another one uses phonemes of the International Phonetic Alphabet, employing an heuristics to transcribe words into phonemes.

Chapter 5

Evaluation methods and datasets

The majority of NLG researches published between 2005-2014 relies on automatic metrics [8]. A lot of NLG evaluations use automatic metrics, 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.

Metrics METEOR, BLEU, ROUGE 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 [17].

In the paper [20] the weak correlation between human and automatic evaluations is also confirmed. Authors compared different word-based (**WBM** relies on ground-truth references) and grammar-based (**GBM** does not rely on ground-truth references) metrics. Their model, combination of WBMs and GBMs, achieved high correlation with humans but only within a single domain.

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. Quality is more important for training data than the quantity of data points. Unfortunately, there are not a lot of datasets available for training NLG models, due to the high cost of creating quality datasets.

In my bachelor thesis I am using 2 different dataset (Twitter data and Persona-Chat).

5.0.1 Persona-Chat dataset

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.

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

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 [41].

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 [41].

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 A.1 and the Figure A.2. An example of Persona-Chat dialogue is shown in the Table 5.1. The dataset contains 262,848 message-responses pairs and splitted into train data - 70%, valid data - 20% and test data - 10%.

5.0.2 Datasets for stylisation and pretraining

Stanford Sentiment Treebank (SST) contains sentences from movie reviews and includes labels for every syntactically plausible phrase in thousands of sentences. To create SST the corpus of movie review excerpts from the paper [22] 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*. Annotators most often used only 5-class classification: negative, somewhat negative, neutral, positive or somewhat positive.

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

Table 5.3: Persona-Chat statistics.

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. A histogram for SST is represented in the Figure A.3.

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 A.4.

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

A dataset of english plaintext jokes was used for generating funny text. 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 A.5.

Twitter dataset contains 867,710 message-response pairs from Twitter and is used for pretraining baseline model. The dataset is splitted into train data - 85%, valid data - 10% and test data - 5%. A histogram for the dataset is shown in the Figure A.6

Chapter 6

Implementation, experiments and evaluations

This chapter describes implemented model for stylized text generation and the experiments performed. The designed architecture is represented in the Figure 6.1. All parts of the model were implemented in Python 3.7.3 with using PyTorch 1.4.0 framework.

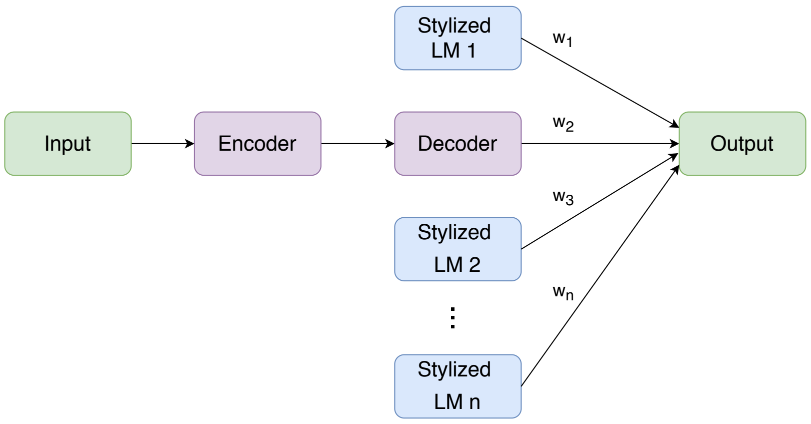

Figure 6.1: An architecture of stylized Natural Language Generation model.

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

The Equation 6.1 is a formal description of the designed model, 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 represented with a softmax over all the words in the vocabulary and multiplied by a corresponding weight w, a hyperparameter to be chosen.

Sequence-to-sequence model was trained on the Persona-Chat dataset, described in the subsection 5.0.1. As the input x to the encoder-decoder model, the entire dialogue history, separated by unique token, is used. Each stylized model was trained on the corresponding stylistic dataset described in the subsection 5.0.2. 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[1]library is used with few handcrafted rules for tokenization.

Beam search, nucleus and temperature sampling were used to generate more varied answers. These decoding methods are described in detail in the section 4.1.

6.1 Baseline

Encoder-decoder model in baseline is implemented as LSTM sequence-to-sequence language model with Luong attention and stylized language model is implemented as LSTM-based model. LSTM neural networks are described in detail in the section 3.3, seq2seq model is described in the section 3.4 and Luong attention is described in the appendix B.

GloVe (Global Vectors) model was used for distributed word representation [24]. It is an algorithm, which 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.

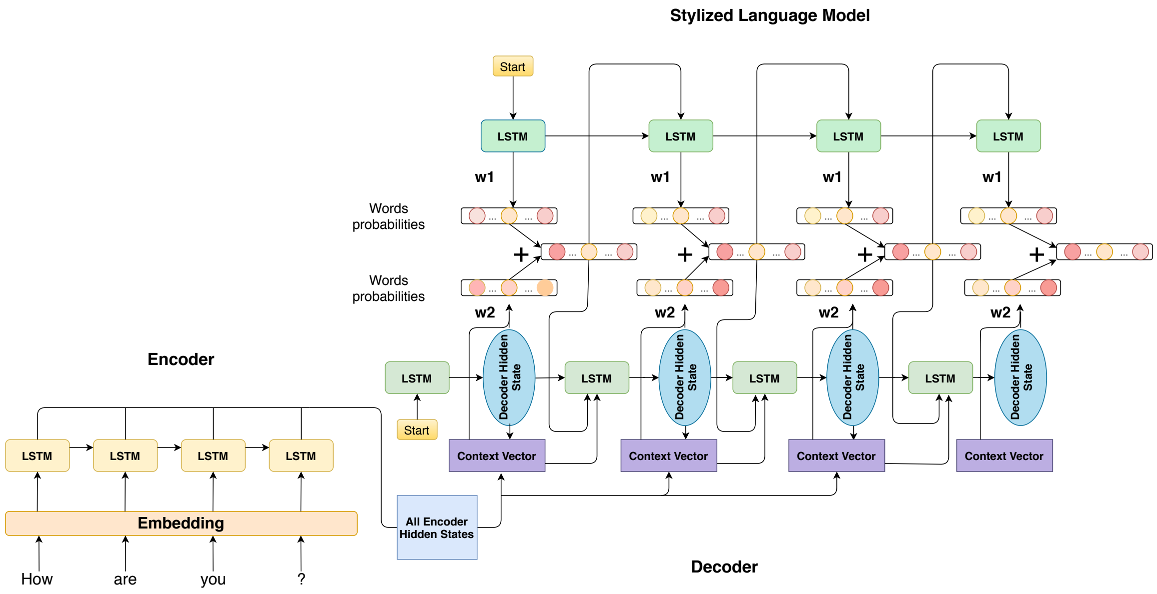

Figure 6.2: Stylized Natural Language Generation baseline model.

6.1.1 Experiment

2 hidden layers were used in both the encoder and decoder with 512 hidden unites in each hidden layer. A dropout [33] 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 6.2. *Adam* [14] is used as the optimization algorithm.

[1]https://spacy.io/

$$H(p,q) = -\sum_{x \in X} p(x)\log(q(x)) \tag{6.2}$$

Equation 6.2: 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.

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 6.1: Example of a dialogue conversation generated by baseline model.

The example, represented in the Table 6.1, shows a repetition problem, such as repetition of the part of an utterance ("I am"), repetition of phrases ("I am afraid"). Another problem is that very often expressions do not make sense in the context. These problems can be because of the small amount of data and LSTM neural networks still has a problem to hold a context. To solve it, I have used transformer-based pretrained models.

6.2 BART and GPT-2

BART and GPT-2 models are used to explore all features of the created architecture (the Figure 6.1). GPT-2 is a successor to GPT, described in the section 3.7 and used as stylized language model. GPT-2 is a direct scale-up of GPT, with more parameters (1.5 billion parameters) and trained on more amount of data. It is a large transformer-based language model, which was trained to predict the next word in 40GB of Internet text.

BART is an autoencoder for pretraining sequence-to-sequence models, described in the section 3.8.

6.2.1 Experiments

Experiment with NIDF

The first experiment was with Normalized Inverse Document Frequency as a decoding feature, described in the section 4.1. This approach makes generated text more specific, because the less often a word is found in a dataset the more weight is setted for the word probability. Example of the generated dialogue is shown in the Table 6.2.

Generated BART conversation.	**Experiment of BART conversation with specifity, weight = 5.**
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?	**Person1:** (Silence)
Person2: that is cool. I am a chef	**Person2:** hello , how are you doing today?
Person1: i don't like carrots either. I am a gourmet cook.	**Person1:** hi! i work as a gourmet cook
Person2: i am very athletic. i love to play sports	**Person2:** what kind of gourmet cook do you cook?
Person1: i ride my bike to work as well.	**Person1:** i do not like carrots. i throw them away.
Person2: that is awesome. i am very athletic, so i love spelling	**Person2:** what kind of carrots do you discard?
Person1: i have been published in a new yorker magazine	**Person1:** really. but, i can sing pitch perfect.
Person2: that is cool. i have never heard of it	**Person2:** what kind of music do you sing pitch perfect?
Person1: i do not make any spelling mistakes	**Person1:** i also cook, and i ride my bike to work.
Person2: that is great!	**Person2:** that sounds fun! i ride my bike to work as well.
Person1: i am from california. my hair is brown	**Person1:** great! i had won an award for spelling bee.
Person2: what is your ethnicity? i am asian.	**Person2:** awesome! what kind of spelling bee did you win?

Table 6.2: Comparison of 2 generated dialogues.

Experiment with weighted decoding

Examples represented in the Tables 6.3, 6.4 shows the generated stylized dialogues with using stylistic datasets.

Experiment of switching models

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:	(Silence)
Person 2:	hey! how are you today?
Person 1:	hi! i work as a gourmet cook
Person 2:	i was just about to go to a gourmet cook shop.
Person 1:	i do not like carrots. i throw them away.
Person 2:	i am a chef for carrots.
Person 1:	really. but, i can sing pitch perfect.
Person 2:	i love the chorus.
Person 1:	i also cook, and i ride my bike to work.
Person 2:	i love riding my bike.
Person 1:	great! i had won an award for spelling bee.
Person 2:	that is awesome. i am sure!

Table 6.3: Experiment BART with GPT-2 trained on dataset with jokes.

Person 1 description: as a child, i won a national spelling bee. # i have been published in the new yorker magazine. # i am a gourmet cook. # i have perfect pitch.	
Person 2 description: i am very athletic. # i wear contacts. # i have brown hair. # i love bicycling. # 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 6.4: Experiment BART and GPT2 with Shakespear style.

Chapter 7

Conclusion

The goal was to create a model that generates a stylistic response to the utterance of an opponent in the dialogue.

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] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] Guu, K., Hashimoto, T. B., Oren, Y. and Liang, P. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics.* MIT Press. 2018, vol. 6, p. 437–450.

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

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

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

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

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

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] Tran, V.-K. and Nguyen, L.-M. Neural-based natural language generation in dialogue using rnn encoder-decoder with semantic aggregation. *ArXiv preprint arXiv:1706.06714.* 2017.

[38] 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.

[39] 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.

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

[41] 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.

[42] 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.

Appendix A

Datasets statistics

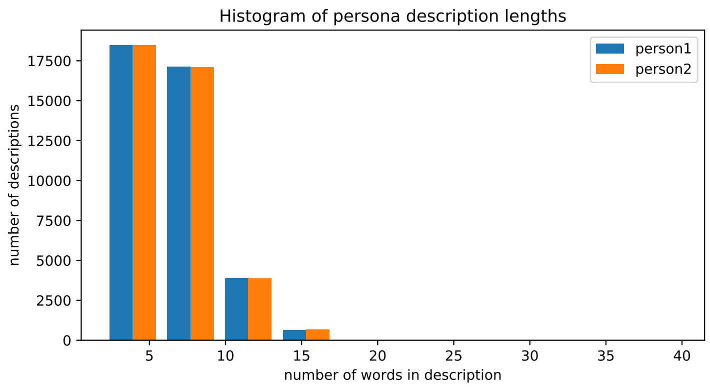

Figure A.1: Histogram of persona description lengths.

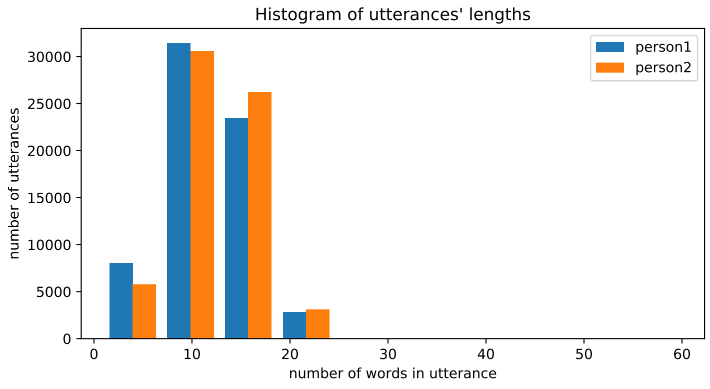

Figure A.2: Histogram of utterances' lengths of a person.

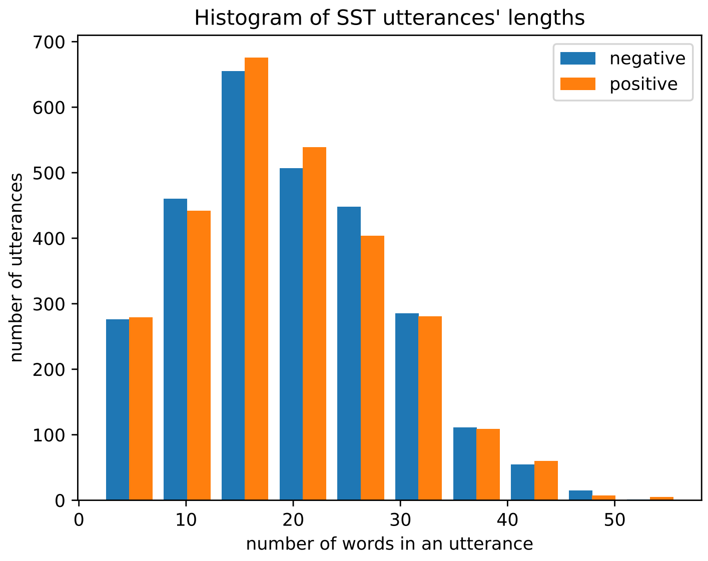

Figure A.3: Histogram of Stanford Sentiment Treebank labeled reviews.

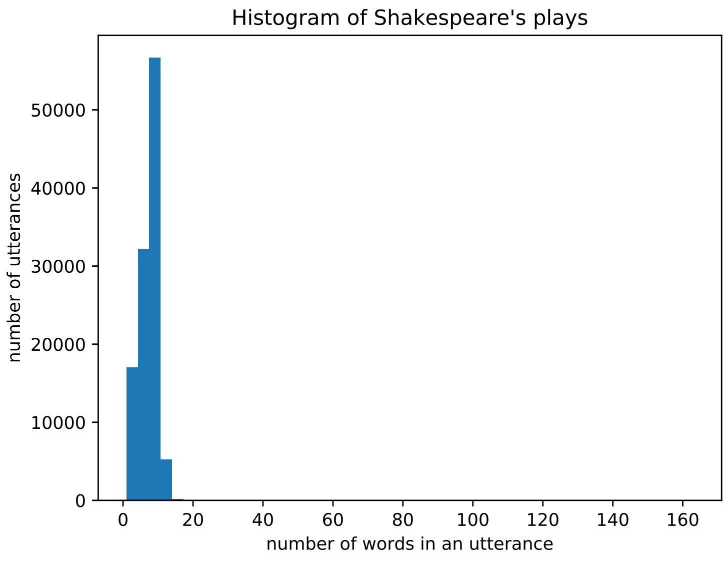

Figure A.4: Histogram of Shakespeare's plays.

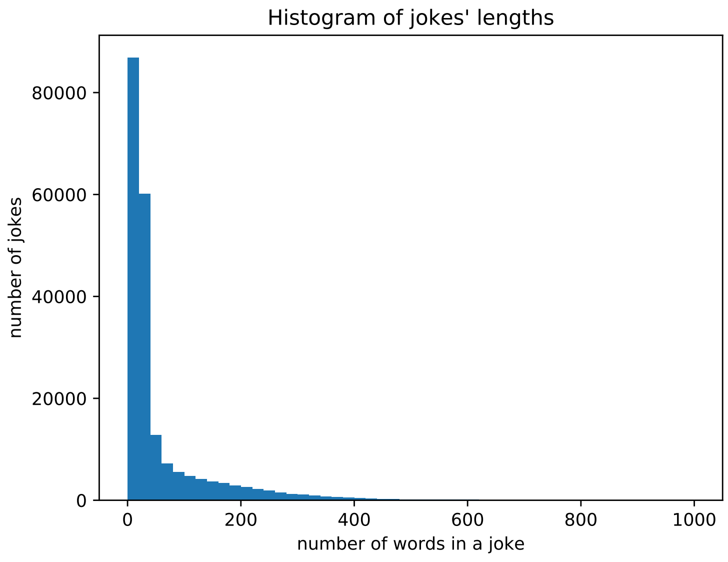

Figure A.5: Histogram of jokes lengths.

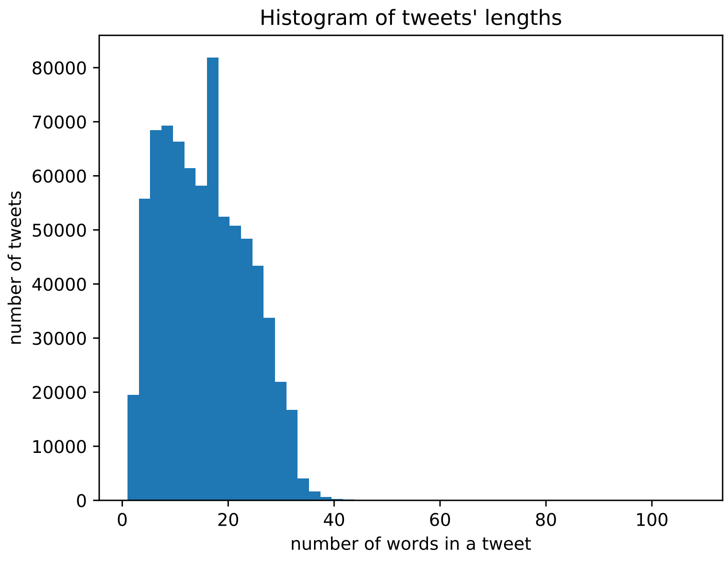

Figure A.6: Histogram of tweets' lengths.

Appendix B

Luong attention

Information in this chapter is taken from [18].

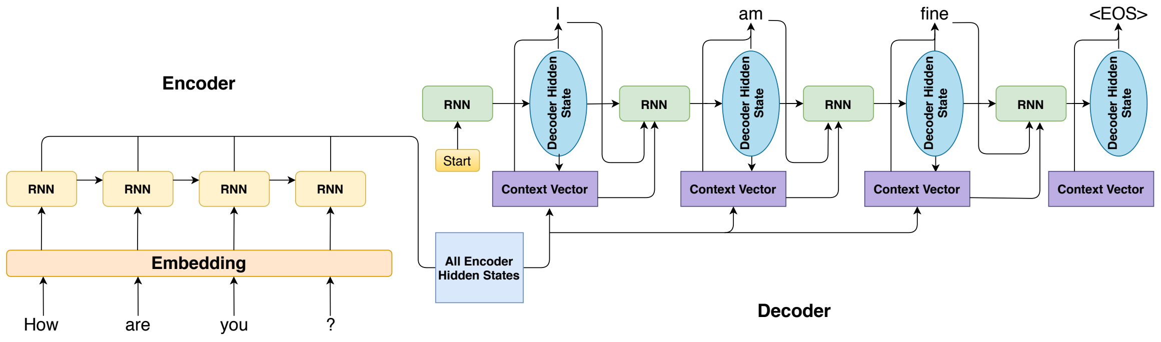

Figure B.1: Luong decoder architecture.

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 B.1).

$$\widetilde{\mathbf{h_t}} = tanh(\mathbf{W_c}[\mathbf{c_t}; \mathbf{h_t}]) \tag{B.1}$$

The attention vector $\widetilde{\mathbf{h_t}}$ then passed through the softmax layer to produce the predictive distribution (Equation B.2).

$$p(y_t|y_{<t}, x) = softmax(\mathbf{W_s}\widetilde{\mathbf{h_t}}) \tag{B.2}$$

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 B.3).

$$a_t(s) = align(\mathbf{h_t}, \bar{\mathbf{h}}_\mathbf{s}) = \frac{exp(score(\mathbf{h_t}, \bar{\mathbf{h}}_\mathbf{s}))}{\sum_{s'} exp(score(\mathbf{h_t}, \bar{\mathbf{h}}_\mathbf{s}))} \tag{B.3}$$

There are three types of the score function (the score function is referred as a content-based function) (Equation B.4).

$$score(\mathbf{h_t}, \mathbf{\bar{h}_s}) = \begin{cases} \mathbf{h_t}^\intercal \mathbf{\bar{h}_s}, & \text{dot} \\ \mathbf{h_t}^\intercal \mathbf{W_a} \mathbf{\bar{h}_s}, & \text{general} \\ \mathbf{v_a}^\intercal tanh(\mathbf{W_a}[\mathbf{h_t}; \mathbf{\bar{h}_s}]), & \text{concat} \end{cases} \tag{B.4}$$

In location-based function the alignment scores are computed from solely the target hidden state $\mathbf{h_t}$ (Equation B.5).

$$a_t = softmax(\mathbf{W_a h_t}) \tag{B.5}$$

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 B.6) and *predictive* (Equation B.7).

$$p_t = t \tag{B.6}$$

$$p_t = S \cdot sigmoid(\mathbf{v_p}^\intercal tanh(\mathbf{W_p h_t})) \tag{B.7}$$

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 B.8).

$$a_t(s) = align(\mathbf{h_t}, \mathbf{\bar{h}_s}) exp(-\frac{(s - p_t)^2}{2\sigma^2}) \tag{B.8}$$