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

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
February 26, 2020

Acknowledgements

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

Contents

1 Introduction **2**

2 NLG problems in dialogue systems **4**
2.1 Template-based approach 4
2.2 Corpus-based approach 5
2.3 Language Models 5
2.4 Dialogue systems 7

3 Models for Natural Language Generation **9**
3.1 Neural Networks 9
3.2 Recurrent neural network (RNN) 10
3.3 Long short-term memory (LSTM) 11
3.4 Sequence-to-sequence model (seq2seq) 12
3.5 Attention 13
3.6 Transformer 13

4 Related works **15**
4.1 Approaches for dialogue systems problems 15
4.2 Approaches for generating stylized and emotional conversation 17

5 Datasets and Evaluation methods **20**
5.0.1 Persona-Chat 20
5.0.2 Twitter 21

6 Conclusion **23**

Bibliography **24**

A Luong attention **26**

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, vendor-specific solutions), 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 [14] 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 an adequate, 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.

In [10] language style is defined as a method of goal-oriented choice and arrangement (organisation) of language means which is applied in the making of the text; in the final product it is thus reflected as the principle of organizing language units which, out of parts and details, shapes a unity compatible with the communicative intention of the author. 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 [11].

2.1 Template-based approach

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.

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 [9]. Example of template-based system is shown in the Table 2.1.

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 [13].

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.

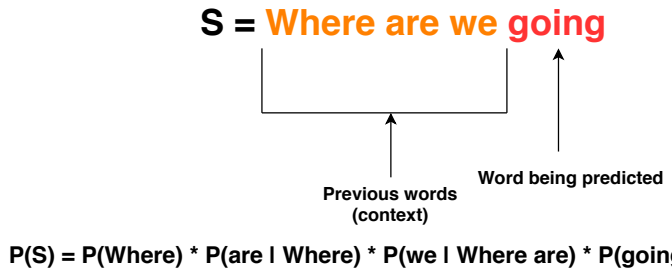

Figure 2.1: Example of a word prediction by using chain rule.

An example in the Figure 2.1 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).

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.

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 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 [2] 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 [4] 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.

-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.

As noticed in [17], 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:

- **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))

-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.

- **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.

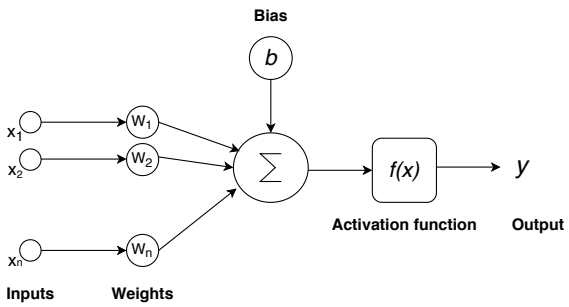

Figure 3.1: Architecture of an artificial neuron.

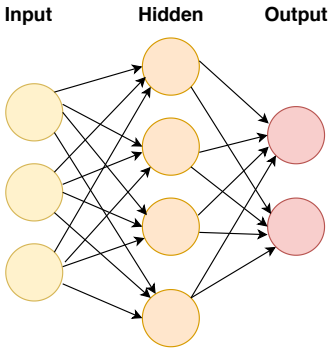

Figure 3.2: The fully-connected neural network with 1 hidden layer.

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.

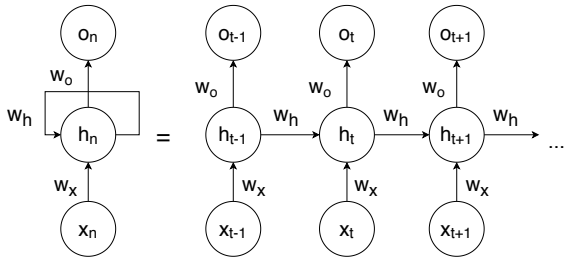

Figure 3.3: Architecture of a recurrent neural network, where coefficient **h** is a hidden state, **x** is an input and **o** is an output. Coefficient **w** is a weight, what is transformed to produce a sensible output.

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 [22] and a training model with semantic aggregation [20].

Nowadays traditional RNN networks almost are not used in NLG, because they have problems with vanishing and exploding gradients. As introduced in [3] 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 [6].

3.3 Long short-term memory (LSTM)

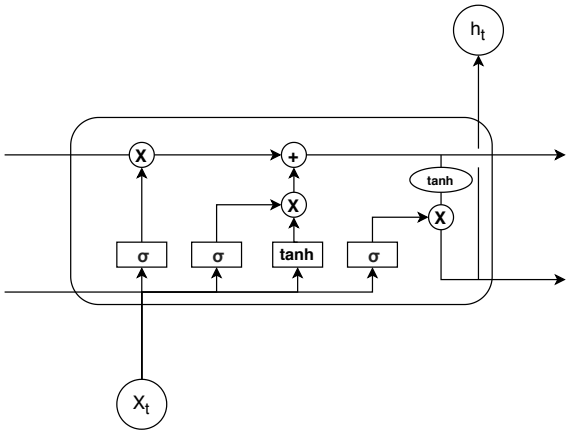

Figure 3.4: A cell in an LSTM network.

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 [18]. 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.

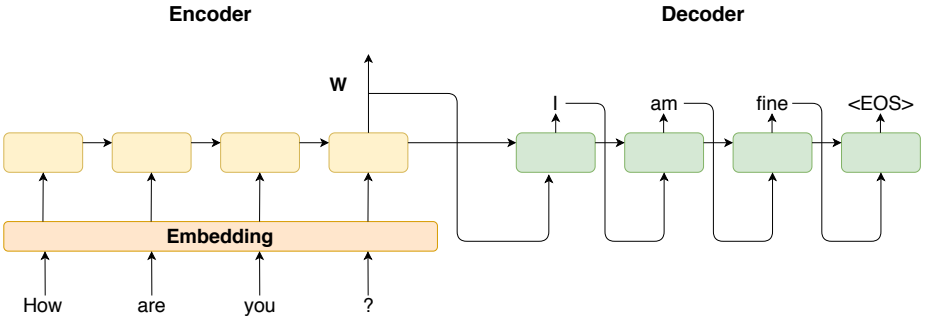

Figure 3.5: Architecture of sequence-to-sequence model.

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 [21] 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 [21].

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

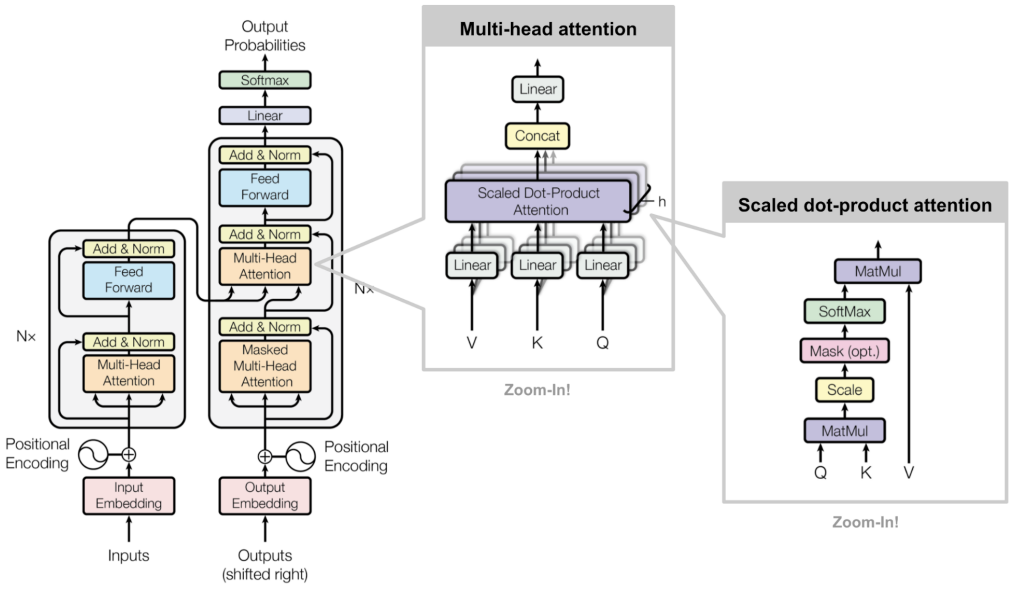

Figure 3.7: The architecture of the Transformer.[1]

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 matricies.

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

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 more emotional and stylized text, which helps make text more human.

4.1 Approaches for dialogue systems problems

In the paper [15] described solutions to common NLG problems in dialogue systems. 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)** [12] and **Weighted Decoding (WD)** [5].

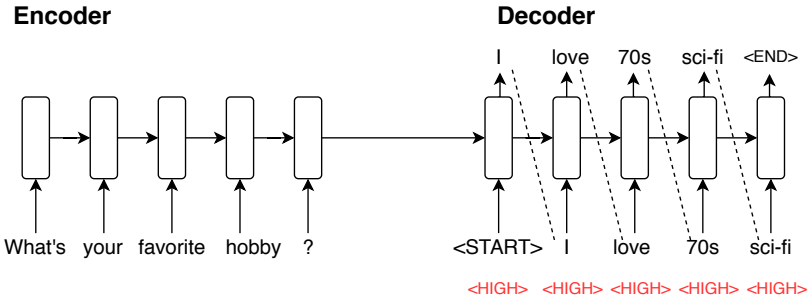

Figure 4.1: Example of Condition Training model.

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.

$$NIDF(w) = \frac{IDF(w) - min\_idf}{max\_idf - min\_idf} \tag{4.1}$$

Normalized Inverse Document Frequency (NIDF) is used as a measure of word rareness (as z variable) for condition training (Equation 4.1). $IDF(w) = \log(\frac{R}{c_w})$ 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.

Question-asking problem is resolved 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 $\frac{i}{10}$.

A **WD** is a technique applied during decoding to increase/decrease the probability of words with certain features. In the paper Weighted Decoding is used for controling specificity by increasing the probability of rare words. 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.

$$score(w, y_{<t}; x) = score(y_{<t}; x) + \log P_{RNN}(w|y_{<t}, x) + \sum_i w_i * f_i(w; y_{<t}, x) \tag{4.2}$$

In WD, 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 (Equation 4.2). $\log P_{RNN}(w|y_{<t}, x)$ 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 $y_{<t}$. $f_i(w; y_{<t}, x)$ are decoding features with associated weights w_i (hyperparameters to be chosen). Each feature presents a specific controlling attribute.

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 is also controled with weighted decoding and 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.

According to [7] decoding strategies with likelihood maximazing lead to text that is increadibly degenerate, even when using state-of-the-art models. This models generate repetitive and overly generic text. 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.

4.2 Approaches for generating stylized and emotional conversation

Affect Listeners

The concept and the constitution of Affect Listeners were introduced in the paper [16]. Affect listeners can respond to users' utterances both at the content- and affectrelated level. This system is able to detect and classify a user's textual expression of affective states and direct a dialog that facilates recognition of the user's topics of interest.

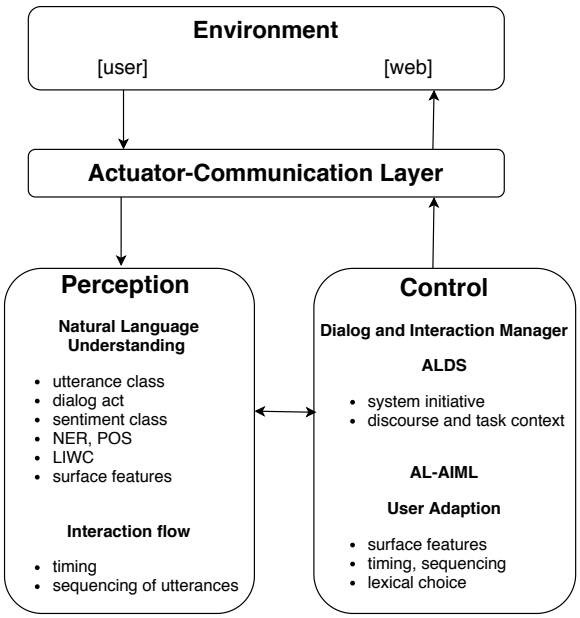

Figure 4.2: Layers of the system architecture.

The architecture, represented in the Figure 4.2, includes 3 layers: perception, control and actuator-communication. Perception layer analysis the input from the system's environment and the output of tools in the actuator layer, by using NLP tools and machine learning-based classifiers. Control layer analysis information from the perception layer and manages interaction with the user. It also monitors the dialog progression and selects a response from the number of response candidates. Affect Listener Dialog Scripting (ALDS), command interpreters for Artificial Intelligence Markup Language (AIML) and User Adaptation Mechanism integrates the rule-based action selection. The actuator-communication layer provides information for generating system responses. This architecture is good for task-oriented dialogues, but for open-ended dialogues it cannot be used, because it is hard to implement all possible scenarios.

MECS

According to [23] in emotional interaction analysis it was found that people tend to achieve the same emotional state in conversation. This phenomenon is explained as empathy at the psychological level.

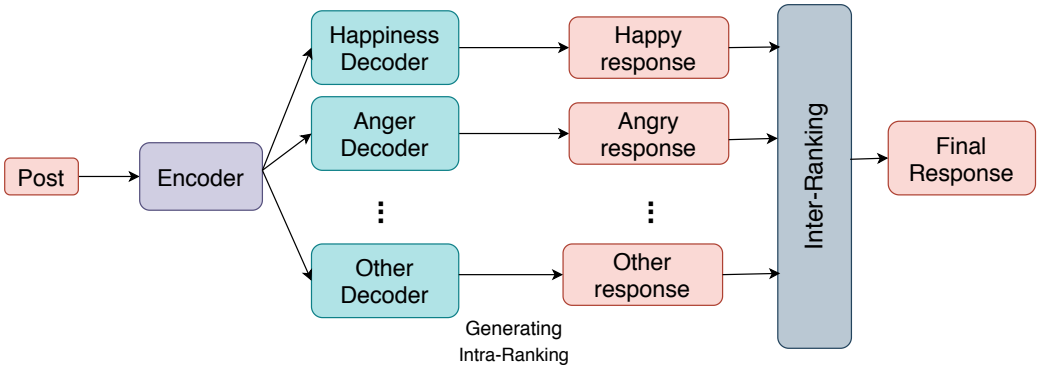

Figure 4.3: The general framework and dataflow of MECS.

In their research they create the system MECS for the NLPCC 2017 shared task on emotional conversation generation. Seq2seq architecture is used to capture the textual information of post sequence and generate responses for each type of emotions simultaneously.

$$p(Y|X) = \prod_{t=1}^{N_y} p(y_t|x_1, x_2, ..., x_T, y_1, y_2, ..., y_{t-1}) = \prod_{t=1}^{N_y} \frac{exp(f(s_t, e_{y_t}))}{\sum_{y'} exp(f(s_t, e_{y'}))} \tag{4.4}$$

The Figure 4.3 represents the architecture of MECS model, where **intra-ranking** and **inter-ranking** are used, to help choosing only one reponse from all generated emotional responses. Calculation of intra-ranking is shown in the Equation 4.4. N_y denotes the length of a response sequence Y, $f(s_t, e_{y_t})$ denotes the corresponding output on the last projection layer for word y_t. The sentence with the highest score is selected as the decoder's output. Inter-ranking selects the most appropriate response as model's output. However in MECS model the inter-ranking is calculated as the itra-ranking (the Equation 4.4), what means that chosen output does not depend on emotional type of input text. Therefore, it is not capable to simulate the emotion interaction in human conversation.

ECM

Emotional Chatting Machine (ECM), represented in [25], 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.

In the architecture (the Figure 4.4) 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

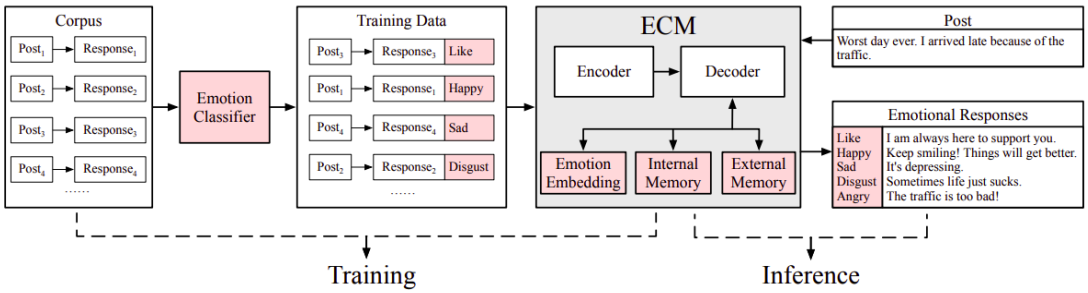

Figure 4.4: Overview of ECM (the grey unit). The pink units are used to model emotion factors in the framework.

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 [19] authors describe the problem of stylized text generation in a multilingual setup. They present a version of a language model based on a LSTM artificial neural network with extended phonetic and semantic embeddings for stylized poetry generation.

$$G(C|S) = \begin{cases} (C, \mathbb{R}^m, F) \to \{T_i^G\} \\ \{T_i^G|S\} \sim \{T_i|S\} \qquad \text{w.r.t. D} \end{cases} \tag{4.5}$$

The equation 4.5 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 and a poetic text are used as a condition S.

Chapter 5

Datasets and Evaluation methods

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

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 [24].

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

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 [24].

Average length of your persona description:	6.332
Average length of partner's persona description:	6.321
Average length of the first person's utterances:	11.419
Average length of the second person's utterances:	11.929
Number of your persona description's sentences:	40239
Number of partner's 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.

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 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. An example of Persona-Chat dialogue is shown in the Table 5.1.

5.0.2 Twitter

Person 1	Person 2
yeah i'm preparing myself to drop a lot on this man, but definitely need something reliable	yeah dude i would definitely consider a daniel defence super reliable and they are just bad ass
besides if trump say his condolences it won't sound genuine, ex: (dwayne wade cousin) it will sound all political and petty	yea you right. but we do live in a world where republicans will harass obama about a birth certificate but won't say

Table 5.4: Example of Twitter message-response pairs

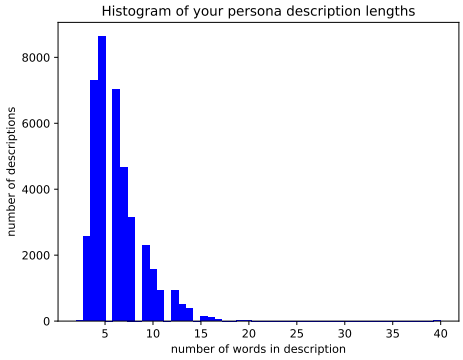

Figure 5.1: Histogram of your persona description lengths.

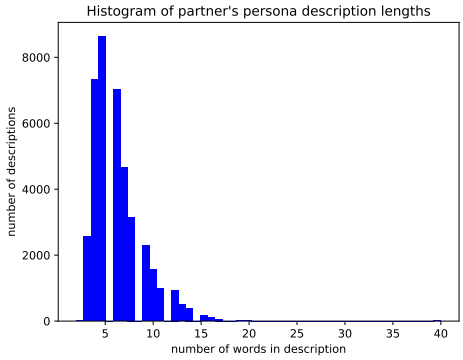

Figure 5.2: Histogram of partner's persona description lengths.

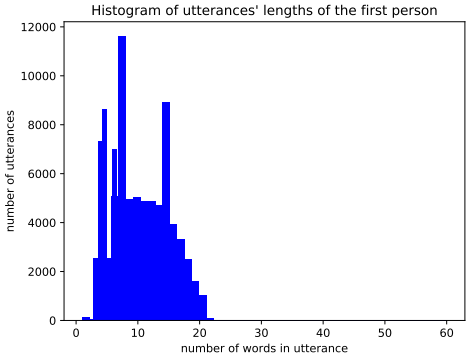

Figure 5.3: Histogram of utterances' lengths of the first person.

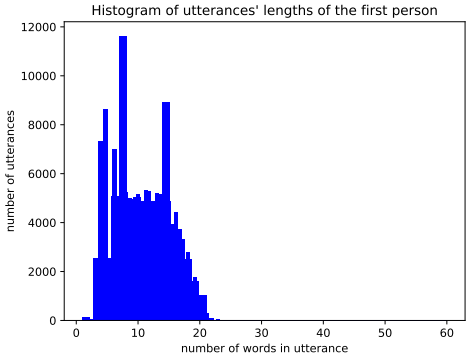

Figure 5.4: Histogram of utterances' lengths of the second person.

Twitter dataset contains message-response pairs from Twitter. An example of this data is presented in the Table 5.4.

Chapter 6

Conclusion

Non-task-oriented dialogue system

The aim of task-oriented dialogue systems is to complete specific tasks fo user, non-task-oriented dialogue systems focus on conversing with human on open domains.

//TODO: Info about datasets

//TODO: Info about evaluation metrics

//TODO: NLP vs Computation linguistic

//Implementation (describe preprocessing, baseline)

Bibliography

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

[2] 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.

[3] 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.

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

[5] 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.

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

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

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

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

[10] Minářová, E., Krčmová, M., Chloupek, J. and Čechová, M. Současná česká stylistika. ISV nakladatelství. 2003.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] Skowron, M. Affect listeners: Acquisition of affective states by means of conversational systems. In: *Development of Multimodal Interfaces: Active Listening and Synchrony.* Springer, 2010, p. 169–181.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] Zhang, R., Wang, Z. and Mai, D. Building emotional conversation systems using multi-task seq2seq learning. In: Springer. *National CCF Conference on Natural Language Processing and Chinese Computing.* 2017, p. 612–621.

[24] 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.

[25] 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

Luong attention

Information in this chapter is taken from [8].

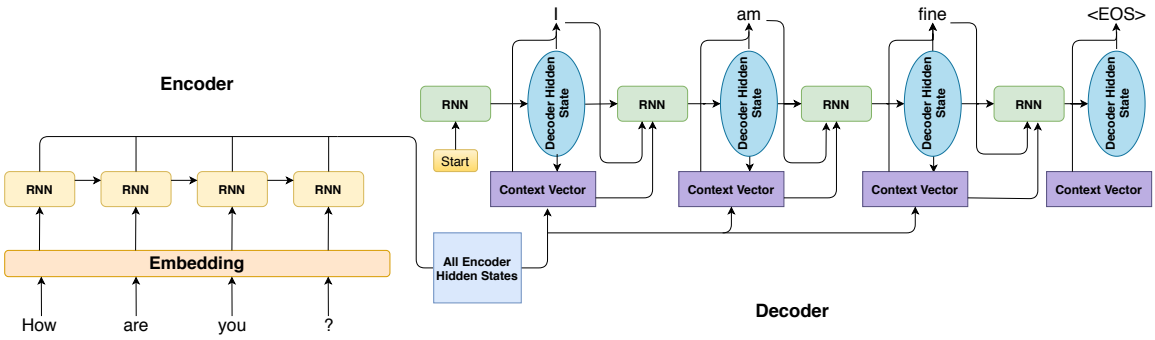

Figure A.1: Luong decoder architecture.

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 A.1).

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

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

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

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 A.3).

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

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

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

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

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

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

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

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

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 A.8).

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