MŰEGYETEM 1782

Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Automation and Applied Informatics

Deep Learning Based Chatbot Models

Scientific Students' Associations Report

Author:
Richárd Krisztián Csáky

Supervised by
Gábor Recski

2017

Kivonat

A konverzációs ágens (chatbot) egy olyan program, mely természetes nyelvet használva képes emberekkel kommunikálni. A beszélgetés modellezése fontos feladat a természetes nyelvfeldolgozás és mesterséges intelligencia (MI) területén. Az MI tudományág megszületése óta egy jól működő chatbot létrehozása még mindig az egyik legnehezebb kihívás. A chatbotok sokféle feladatra használhatók, de mindegyik esetében elvárt, hogy megértsék a felhasználó mondandóját és az adott problémához releváns válaszokat generáljanak.

A múlt chatbot architektúrái kézi szabályokra és sablonokra, vagy egyszerű statisztikai módszerekre támaszkodtak. 2015 óta, a mélytanulás (deep learning) elterjedésével ezek a modellek gyorsan felcserélődtek elejétől végéig tanítható neurális hálózatokkal. Manapság a rekurrens enkóder-dekóder modell [Cho et al., 2014] dominál a konverzáció modellezésben. Ezt az architektúrát a neurális gépi fordítás területéről adaptálták, ahol rendkívül jó eredményeket ért el. Azóta sokféle változata [Serban et al., 2016] és kiegészítése született annak érdekében, hogy minél jobb minőségű legyen a chatbotok által folytatott beszélgetés.

Munkám során részletes irodalmi kutatást végeztem, melyben az elmúlt 3 évben publikált, több mint 70, a chatbotokkal kapcsolatos publikációt vizsgálok meg. Ezután amellett érvelek, hogy a konverzáció modellezés sajátosságai a jelenlegi state-of-the-art architektúráktól eltérő megközelítést igényelnek. Szakirodalmi példákon alapulva bemutatom, hogy a jelenlegi chatbot modellek miért nem vesznek figyelembe elég ún. priort a válasz generálása során, és ez hogyan befolyásolja a beszélgetés minőségét. Ezek a priorok olyan külső információt hordoznak, melyen a beszélgetés kondicionálva lehet, mint például a beszélők személye [Li et al., 2016a] vagy hangulata. Amellett, hogy bemutatom az okait, javaslatokat is teszek a probléma orvoslására.

A dolgozat következő részében egy nemrég bemutatott modellt, mely jelenleg state-of-the-art-nak számít a neurális gépi fordításban, az úgynevezett Transformer-t [Vaswani et al., 2017] adaptálom a beszélgetés-modellezés feladatára. Először az eredeti cikkben leírt modell tanításával kísérletezek, tanítóadatként a Cornell Movie-Dialog Corpus [Danescu-Niculescu-Mizil and Lee, 2011] dialógusait használva. Emellett továbbfejlesztem a modellt saját, az enkóder-dekóder architektúra hiányainak orvoslására született ötletekkel. További priorokat adok bemenetként a modellbe, mint a beszélgetők személye vagy hangulata. Végül korábbi chatbot modellekkel való összehasonlítás útján részletes elemzést végzek arról, hogy az eredeti modell mennyire teljesít jól dialógus adattal és hogyan befolyásolják a generált válaszok minőségét az általam implementált további kiegészítések.

Abstract

A conversational agent (chatbot) is a piece of software that is able to communicate with humans using natural language. Modelling conversation is an important task in natural language processing and artificial intelligence (AI). Indeed, ever since the birth of AI, creating a good chatbot remains one of the field's hardest challenges. While chatbots can be used for various tasks, in general they have to understand users' utterances and provide responses that are relevant to the problem at hand.

In the past, methods for constructing chatbot architectures have relied on hand-written rules and templates or simple statistical methods. With the rise of deep learning these models were quickly replaced by end-to-end trainable neural networks around 2015. More specifically, the recurrent encoder-decoder model [Cho et al., 2014] dominates the task of conversational modelling. This architecture was adapted from the neural machine translation domain, where it performs extremely well. Since then a multitude of variations [Serban et al., 2016] and features were presented that augment the quality of the conversation that chatbots are capable of.

In my work, I conduct an in-depth survey of recent literature, examining over 70 publications related to chatbots published in the last 3 years. Then I proceed to make the argument that the very nature of the general conversation domain demands approaches that are different from current state-of-the-art architectures. Based on several examples from the literature I show why current chatbot models fail to take into account enough priors when generating responses and how this affects the quality of the conversation. In the case of chatbots these priors can be outside sources of information that the conversation is conditioned on like the persona [Li et al., 2016a] or mood of the conversers. In addition to presenting the reasons behind this problem, I propose several ideas on how it could be remedied.

The next section of my paper focuses on adapting the very recent Tranformer [Vaswani et al., 2017] model to the chatbot domain, which is currently the state-of-the-art in neural machine translation. I first present my experiments with the vanilla model, using conversations extracted from the Cornell Movie-Dialog Corpus [Danescu-Niculescu-Mizil and Lee, 2011]. Secondly, I augment the model with some of my ideas regarding the issues of encoder-decoder architectures. More specifically, I feed additional features into the model like mood or persona together with the raw conversation data. Finally, I conduct a detailed analysis of how the vanilla model performs on conversational data by comparing it to previous chatbot models and how the additional features, affect the quality of the generated responses.

Contents

1 Introduction

A conversational agent (chatbot) is a piece of software that is able to communicate with humans using natural language. Ever since the birth of AI, modelling conversations remains one of the field's toughest challenges. Even though they are far from perfect, chatbots are now used in a plethora of applications like Apple's Siri [Apple, 2017], Google's Google Assistant [Google, 2017] or Microsoft's Cortana [Microsoft, 2017a]. In order to fully understand the capabilities and limitations of current chatbot architectures and techniques I conduct an in-depth survey, where I examine related literature published over the past 3 years and I implement my own chatbot based on a novel neural network model.

The paper begins with a brief overview of the history of chatbots in Section 2, where I discuss the properties and objectives of conversational modelling. I present early approaches and the current dominating model based on neural networks for building conversational agents.

In the following section I present key architectures and techniques that were developed over the past 3 years relating to chatbots. I group publications into 11 groups based on the specific techniques or approaches that are discussed by the authors. After this I present criticism regarding some of the properties of current chatbot models and I show how several of the techniques used are inappropriate for the task of modelling conversations.

In the next section I conduct experiments by training a novel neural network model, the Transformer [Vaswani et al., 2017] using dialog datasets [Danescu-Niculescu-Mizil and Lee, 2011, Tiedemann, 2009, Lison and Tiedemann, 2016]. I run several trainings using these datasets detailed in Section 4. I present the results of the different training setups by qualitatively comparing them to previous chatbot models and by using standard evaluation metrics.

In the final section before concluding I offer possible directions for future work. More specifically, I propose several ideas in order to remedy the problems discussed in Section 3.2.

2 History of Chatbots

2.1 Modelling Conversations

Chatbot models usually take in as input natural language sentences uttered by the user, and output a response. There are two main approaches for generating responses. The traditional approach is to use hard-coded templates and rules to make chatbots, which I present in Section 2.2. The more novel approach, which I discuss in detail in Section 2.3 was made possible by the rise of deep learning. Neural network models are trained on large amounts of data to learn the process of generating relevant and grammatically correct responses to input utterances. Models have also been developed to accommodate for spoken or visual inputs. They oftentimes make use of a speech recognition component to transform speech into text [Serban et al., 2017] or convolutional neural networks that transform the input pictures into useful representations for the chatbot [Havrylov and Titov, 2017]. The latter models are also called visual dialog agents, where the conversation is grounded on both textual and visual input [Das et al., 2017].

Conversational agents exist in two main forms. The first one is the more traditional task-oriented dialog system, which is limited in its conversational capabilities, however it is very robust at executing task specific commands and requirements. Task-oriented models are built to accomplish a specific task like making restaurant reservations [Joshi et al., 2017, Bordes et al., 2016] or promoting movies [Yu et al., 2017], just to name a few. These systems often don't have the ability to respond to arbitrary utterances since they are limited to a specific domain, thus users have to be guided by the dialog system towards the task at hand. Usually they are deployed to tasks where some information has to be retrieved from a knowledge base. They are mainly used to replace the process of navigating through menus and user interfaces like making the process of booking flight tickets or finding a public transportation route between two locations conversational [Zhao et al., 2017].

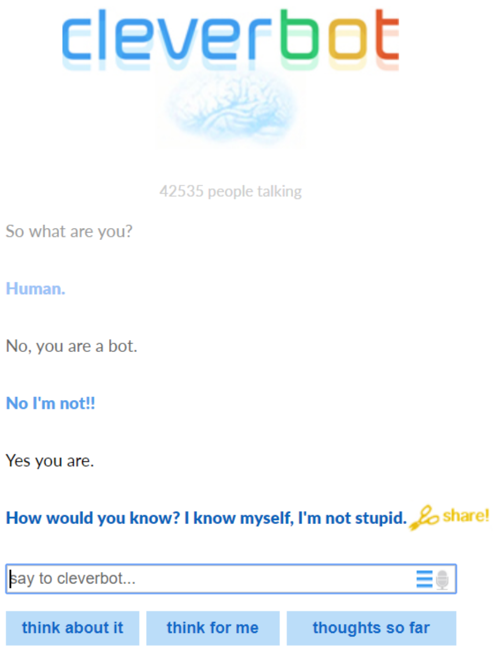
The second type of dialog agents are the non-task or open-domain chatbots. These conversation systems try to imitate human dialog in all its facets. This means that one should hardly be able to distinguish such a chatbot from a real human, but current models are still far away from such claims. These models are usually trained with conversation examples extracted from movie scripts or from Twitter-like post-reply pairs [Vinyals and Le, 2015, Shang et al., 2015, Serban et al., 2016, Li et al., 2016a]. For these models there isn't a well defined goal, but they are required to have a certain amount of world knowledge and commonsense in order to hold conversations about basically anything.

Recently an emphasis has been put on integrating the two types of conversational agents. The main idea is to combine the positive aspects of both types, like the robust abilities of goal-oriented dialog systems to perform tasks and the human-like chattyness of open-domain chatbots [Zhao et al., 2017, Yu et al., 2017, Serban et al., 2017]. This is beneficial because the user is more likely to engage with a task-oriented dialog agent if its more human-like, and handles out of domain responses well.

2.2 Early Approaches

ELIZA is one of the first ever chatbot programs written [Weizenbaum, 1966]. It uses clever hand-written templates to generate a reply that resembles the user's input utterance. Since then countless hand-coded rule-based chatbots have been written [Wallace, 2009, Carpenter, 2017, Worswick, 2017]. Furthermore, a number of programming frameworks specifically designed to facilitate building dialog agents have been developed [Marietto et al., 2013, Microsoft, 2017b].

These chatbot programs are very similar in their core, namely that they all use hand-written rules to generate replies. Usually simple pattern matching or keyword retrieval techniques are employed to handle the user's input utterance. Then rules are used to transform a matching pattern or a keyword into a predefined reply.



So what are you?

Human.

No, you are a bot.

No I'm not!!

Yes you are.

How would you know? I know myself, I'm not stupid.

Figure 1: Sample conversation with Cleverbot [Carpenter, 2017]

2.3 The Encoder-Decoder Model

The main concept that differentiates rule-based and neural network based approaches is the presence of a learning algorithm in the latter case. Instead of using hand-written rules deep learning models transform input sentences into replies directly by using matrix multiplications and non-linearities over millions of parameters. We can further divide neural network based models into two categories, retrieval-based and generative models. The former simply returns a reply from the dataset by computing the most likely response to the current input utterance based on a scoring function, which can be implemented as a neural network [Cho et al., 2014] or by simply computing the cosine similarity between the input utterance and candidate replies [Li et al., 2016b]. Generative models on the other hand synthesize the reply one word at a time by computing probabilities over the whole vocabulary [Sutskever et al., 2014, Vinyals and Le, 2015]. There have also been approaches that integrate the two types of dialog systems by comparing a generated reply with a retrieved reply and determining which one is more likely to be a better response [Song et al., 2016].

As with many other applications the field of conversational modelling has been transformed by the rise of deep learning. More specifically the encoder-decoder recurrent neural network (RNN) model (also called seq2seq [Sutskever et al., 2014]) introduced by [Cho et al., 2014] and its variations have been dominating the field. This model was originally developed for neural machine translation (NMT), but it was found to be suitable to *translate* source utterances into responses within a conversational setting [Shang et al., 2015, Vinyals and Le, 2015].

2.3.1 Recurrent Neural Networks

A recurrent neural network (RNN) [Rumelhart et al., 1988] is a neural network that can take as inputs a variable length sequence $\boldsymbol{x} = (x_1, ..., x_n)$, by unfolding itself over each input x_i and generating a hidden state at each step $\boldsymbol{h} = (\boldsymbol{h_1}, ..., \boldsymbol{h_n})$. At each step i, the hidden state $\boldsymbol{h}_i$ is updated by

$$\boldsymbol{h}_i = f(\boldsymbol{h}_{i-1}, x_i) \tag{1}$$

which is also called the unrolling of the network. f here is a non-linear activation function. Usually long short-term memory (LSTM) or gated recurrent units (GRU) are used for the activation function [Hochreiter and Schmidhuber, 1997, Cho et al., 2014]. An important characteristic of recurrent neural networks is that the parameters of the function f don't change during the unrolling of the network.

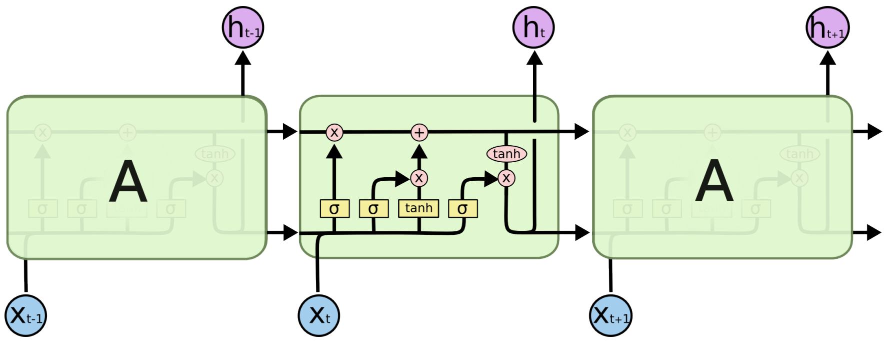


Figure 2: Unrolling an LSTM network over 3 time steps [Olah, 2015]

RNNs can be used for language modelling by training it to learn the probability distribution over the vocabulary $\boldsymbol{V}$ given an input sequence of words. We can generate this probability distribution to predict the next word in the sequence by taking the hidden state of the RNN in the last step, and feeding it into a softmax activation function

$$p(x_{i,j}|x_{i-1}, ..., x_1) = \frac{exp(\boldsymbol{w}_j \boldsymbol{h}_i)}{\sum_{j'=1}^{K} exp(\boldsymbol{w}_{j'} \boldsymbol{h}_i)} \tag{2}$$

for all possible words (symbols) $j = 1, ..., K$, where $\boldsymbol{w}_j$ are the rows in the weight matrix of the softmax function.

Training of these networks is done via the generalized backpropagation algorithm called truncated backpropagation through time [Werbos, 1990, Rumelhart et al., 1988]. Essentially we backpropagate the error through each time step of the network to learn its parameters. The error can be computed by using the cross-entropy loss, which calculates how different the predictions are compared to the true labels.

$$L(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) = -\boldsymbol{y}_i log(\hat{\boldsymbol{y}}_i) \tag{3}$$

where $\hat{\boldsymbol{y}}_i$ is the vector of the predicted probabilities over all words in the vocabulary at step i, and $\boldsymbol{y}_i$ is the one-hot vector over the vocabulary. A one-hot vector is made up of zeros except at the index of the one true word that follows in the sequence, where it is equal to 1. After computing the derivative with respect to all of the weights in the network using the backpropagation through time algorithm we can update the weights in order to get closer to an optimum with optimization techniques like stochastic gradient descent (SGD) [Bottou, 2010].

2.3.2 The Seq2seq Model

The sequence to sequence model (seq2seq) was first introduced by [Cho et al., 2014], but they only used it to re-rank sentences instead of generating completely new ones, which was first done by [Sutskever et al., 2014]. Since then, besides NMT and conversational models a plethora of different applications of these models have been introduced like text summarization [Nallapati et al., 2016], speech recognition [Chiu et al., 2017], code generation [Barone and Sennrich, 2017] and parsing [Konstas et al., 2017], just to name a few.

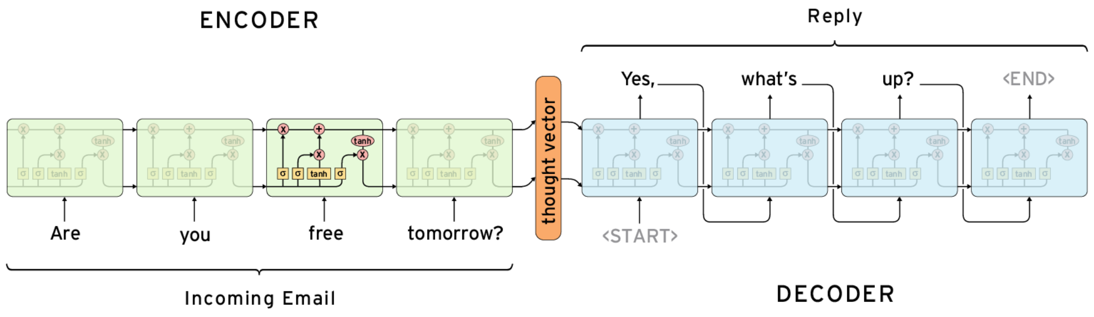


Figure 3: LSTM based Seq2seq model applied to conversational modelling [Britz, 2016]

The simplest and initial form of the model is based on two RNNs. The actual implementation of RNNs can be in the form of LSTMs or GRUs as discussed in Section 2.3.1. The goal is two estimate the conditional probability of $p(y_1, ..., y_{N'}|x_1, ..., x_N)$, where $x_1, ..., x_N$ is the input sequence and $y_1, ..., y_{N'}$ is the corresponding output sequence. Since two different RNNs are used for the input and output sequences, the length of the sequences, N and $N^{'}$ can be different. The encoder RNN is unrolled over the words in the source sequence and its last hidden state is called the thought vector, which is a representation of the whole source sequence. The initial hidden state of the decoder RNN is then set to this representation, and the generation of target sequence words is done by taking the output of the unrolled decoder network at each time-step and feeding it into a softmax function to, which produces the probabilities over all words in the vocabulary.

$$p(y_1, ..., y_{N'}|x_1, ..., x_N) = \prod_{i=1}^{N'} p(y_i|v, y_1, ..., y_{i-1}) \tag{4}$$

Training is very similar to a normal RNN, namely the log probability of a correct target sequence T given the source sequence S is maximized

$$\frac{1}{\boldsymbol{S}} \sum_{T,S \in \boldsymbol{S}} log(p(T|S)) \tag{5}$$

where $\boldsymbol{S}$ is the training set. The two networks are trained jointly, errors are backpropagated through the whole model and the weights are optimized with some kind of optimization technique like SGD.

In NMT the input sequences as sentences in one language from which we wish to translate and target sequences are sentences in a different language to which we wish to translate. In conversational modelling the simplest approach is to treat an utterance by a speaker as input sequence and the response to that utterance from a different speaker as the target sequence. I will talk about better approaches however in Section 3.1.2.

2.3.3 Deep Seq2seq Models

Seq2seq models can also contain multiple layers of LSTM networks as seen in Figure 2.3.3. This is done in order to make the model deeper and to have more parameters, which should ideally lead to better performance [Vinyals and Le, 2015, Wu et al., 2016]. There exist multiple variants, but the most straightforward one is to feed in the source sentence to the first layer of the encoder network. The output from the previous LSTM layer is fed as input to the next layer and the layers are unrolled jointly. Then the last hidden state of the final encoder layer can be used to initialize the initial hidden state of the first decoding layer. The output of the previous decoder layer is input to the next layer until the final layer, where we apply a softmax activation function over the outputs from the last layer to generate the predicted target sequence. How we initialize the layers in the decoder network can be implemented in many different ways, like taking the last hidden state from each encoder layer and using it to initialize the first hidden state of each corresponding decoder layer, to name another method.

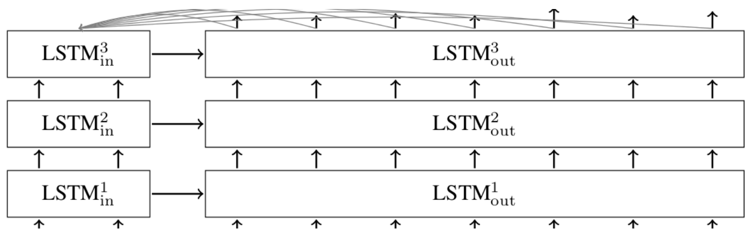

$\text{LSTM}^3_{\text{in}}$
$\text{LSTM}^3_{\text{out}}$
$\text{LSTM}^2_{\text{in}}$
$\text{LSTM}^2_{\text{out}}$
$\text{LSTM}^1_{\text{in}}$
$\text{LSTM}^1_{\text{out}}$

Figure 4: A 3 layer Sequence-to-sequence model [Tensorflow, 2017]. The lines pointing from the last decoder states to the last encoder represent an attention mechanism about which I talk in Section 3.1.1

2.3.4 Decoding and Vocabulary

In order to get the actual generated output sequences there are several techniques to decode from the probabilities of the words. One such technique is the Roulette Wheel selection, which is commonly used for genetic algorithms [Mitchell, 1998]. Using this method at each time-step each word from the vocabulary can be generated with the probability computed from the softmax function. This is useful if we want to have some stochasticity in the decoding process, because it can produce slightly different outputs for the same source sequence. However it doesn't perform as well as the more frequent method used, which is to simply output the word with the highest probability from the softmax function. This is a greedy and deterministic approach since we will always get the same output for the same input.

While decoding a word at each time-step is fine a better approach is to the decode the whole output sequence at the same time, by outputting the sequence with the highest probability.

$$\hat{T} = \arg\max_{T} p(T|S) \tag{6}$$

Since in order to get the sequence with the highest probability we have to first generate all of the possible sequences a simple left-to-right beam search is usually employed to make the computation tractable. First, K words with the highest probability are retained in the first time-step, then at each time-step we expand this list by computing the joint probability of the partial sequences in the list and the words in the current time-step and retaining the K most probable partial sequences until we get to the end of the output sequence.

Another important aspect of sequence-to-sequence models applied to tasks involving language is the vocabulary. The vocabulary consists of all the different words and symbols present in the dataset. One problem with this approach is that the vocabulary tends to be quite large, especially for very big datasets like [Lison and Tiedemann, 2016, opensubtitles.org, 2017]. Since the number of parameters of the model increases proportionally with the size of the vocabulary it is usually the case that the vocabulary is limited to some arbitrary size K. This way we only use the embeddings of the K most frequent words in the dataset and replace any other symbols with a common token representing unknown words. Many approaches have been proposed to the problem of out of vocabulary (OOV) or unknown words [Luong et al., 2014, Feng et al., 2017, Jean et al., 2014]. Other methods involve using characters [Zhu et al., 2017] or subword units [Sennrich et al., 2015] instead of words.

3 Background

In this section I will first present selected publications from my literature research. I categorize these papers into several categories, each corresponding to a specific approach or aspect of conversational modelling. Then I present some criticism regarding basic techniques used in neural conversation models. While they are widely used, I show that the assumptions on which their usage rests are generally wrong and in consequence these methods are unsuitable for modelling conversations. Finally I provide a brief summary of the section.

3.1 Recent Chatbot Architectures and Augmentations

In this section I present recent methods and techniques used for neural networks based conversation models. I describe different attention mechanism and how they can benefit encoder-decoder models. Furthermore I show how context and previous utterances in conversations can be taken into account with different techniques. Then I talk about different objective functions which have been proposed to combat conversation specific issues. I also describe a number of papers where additional features were used as input for encoder-decoder models and how can knowledge bases and information retrieval be integrated into conversational models. I present some approaches to integrate task-oriented conversations and goals with encoder-decoder models. Also reinforcement learning is a new approach which has seen some success when applied to training conversational agents. Finally, I present encoder-decoder models that are very different from the standard RNN based seq2seq, but nonetheless have achieved state-of-the-art results and I conclude with describing the evaluation methods that exist for measuring the performance of neural conversational models.

3.1.1 Attention

The attention mechanism was first introduced to encoder-decoder models by [Bahdanau et al., 2014]. The problem that it was trying to address is the limited information that the context vector can carry. In a standard seq2seq model we expect that this single fixed-size vector can encode all the relevant information about the source sentence that the decoder needs in order to generate its prediction. Additionally, since we use a single vector, all the decoder states receive the same information, instead of feeding in information relevant to the specific decoding step.

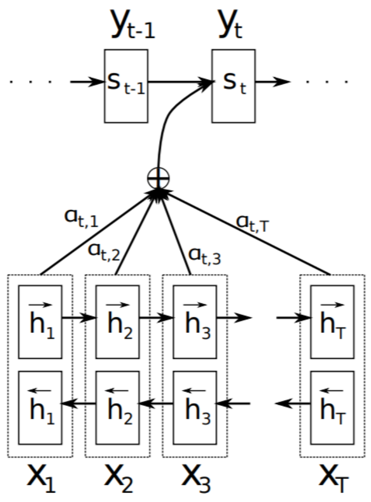


Figure 5: Original Attention Mechanism [Bahdanau et al., 2014]

In order to combat these shortcomings the attention mechanism creates an additional input at each decoding step coming directly from the encoder states. This additional input c_i to the decoder at time-step i is computed by taking a weighted sum over the encoder hidden states $\boldsymbol{h}$:

$$c_i = \sum_{j=1}^{T} a_{ij} h_j \tag{7}$$

where T is the number of hidden states or symbols in the source sentence and the weight a_{ij} for each h_j hidden state can be computed by

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T} exp(e_{ik})} \tag{8}$$

which is basically a softmax function over e_{ij}, which is the output of a scoring function:

$$e_{ij} = f(s_{i-1}, h_j). \tag{9}$$

Here s_{i-1} is the hidden state of the decoder in the previous time-step. Oftentimes s_{i-1} is called a query vector $\boldsymbol{q}$ and the encoder hidden states h are called key vectors $\boldsymbol{k}$. The scoring function f can be implemented in several ways:

$$MLP(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{w_2} tanh(W_1[\boldsymbol{q} : \boldsymbol{k}]) \tag{10}$$

where we simply concatenate the query and key vectors and feed them into a single or multilayer neural network [Bahdanau et al., 2014].

$$BL(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^{\mathsf{T}} W \boldsymbol{k} \tag{11}$$

is called the bilinear scoring function [Luong et al., 2015].

$$DP(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^{\mathsf{T}}\boldsymbol{k} \tag{12}$$

is the dot product scoring function, which doesn't use any weights, but it requires the sizes of the two vectors to be the same [Luong et al., 2015].

$$SDP(\boldsymbol{q}, \boldsymbol{k}) = \frac{\boldsymbol{q}^{\mathsf{T}}\boldsymbol{k}}{\sqrt{|\boldsymbol{k}|}} \tag{13}$$

is the scaled dot product, where we scale the previous dot product by the square root of the size of the key vectors [Vaswani et al., 2017]. This is useful because otherwise the dot product increases as dimensions get larger.

We can attend to outputs from the decoder network generated in previous time-steps by incorporating them as additional inputs to the scoring functions mentioned above [Shao et al., 2017]. This helps the decoder by allowing it to *see* what has already been generated.

Attention can also be implemented using multiple heads [Vaswani et al., 2017], meaning that we use the output of multiple scoring functions, each with their own parameters to learn to focus on different parts of the input sequence. The parameters of the scoring functions are jointly learned with all other parameters of the seq2seq model.

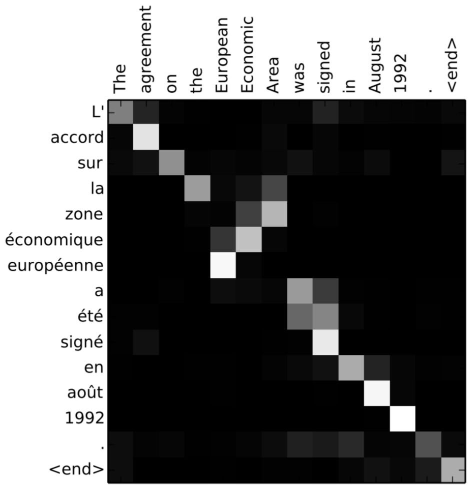


Figure 6: Pixels represent the weights a_{ij} between inputs and outputs [Bahdanau et al., 2014]

Attention can also be computed using only one sequence of symbols [Cheng et al., 2016, Lin et al., 2017, Vaswani et al., 2017]. In this case the query and key vectors both come from the same hidden states and thus we get an encoding of the sentence which depends on all of its parts. This is called self or intra-attention and it can be used both in the decoder and the encoder layer since it can be implemented as a standalone layer that takes in a sequence and computes a new representation of that sequence.

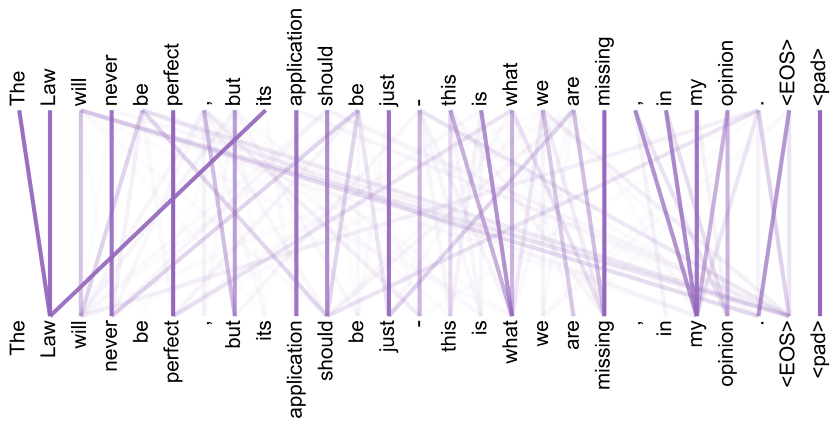


Figure 7: Visualizing self-attention, lines represent the weights a_{ij} [Vaswani et al., 2017]

Attention has been extensively used for neural conversation models as an augmentation of the base seq2seq model [Yao et al., 2016, Shang et al., 2015, Xing et al., 2017a, Zhao et al., 2017]. A more complex type of attention mechanism is the hierarchical attention which was used in conversational modelling [Xing et al., 2017b] and abstractive text summarization [Nallapati et al., 2016] as well. It is useful when we need to attend over multiple input sentences. I describe it in more detail in Section 3.1.3.

3.1.2 Context

3.1.3 Hierarchical Models

3.1.4 Objective Functions

3.1.5 Additional Features and Priors

3.1.6 Pretraining

3.1.7 Knowledge Bases, Copying and Information Retrieval

3.1.8 Task-Oriented Approaches

3.1.9 Reinforcement Learning

3.1.10 Techniques to Augment Seq2seq Models

3.1.11 Different Encoder-Decoder Models

3.1.12 Evaluation Methods

3.2 Criticism

3.2.1 Datasets

3.2.2 The Loss Function

3.2.3 Memory

3.2.4 Evaluation Metrics

3.3 Summary

4 Experiments

4.1 The Tranformer Model

4.2 Datasets

4.3 Training Details

5 Results

5.1 Quantitative Analysis

5.2 Qualitative Analysis

6 Future Work

6.1 Ideas Towards Solving The Loss Function Issue

6.2 Temporal Conditioning and Memory

6.3 Additional Ideas

7 Conclusion

References

[Apple, 2017] Apple (2017). Siri. `https://www.apple.com/ios/siri/`. Accessed: 2017-10-04.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Barone and Sennrich, 2017] Barone, A. V. M. and Sennrich, R. (2017). A parallel corpus of python functions and documentation strings for automated code documentation and code generation. *arXiv preprint arXiv:1707.02275*.

[Bordes et al., 2016] Bordes, A., Boureau, Y.-L., and Weston, J. (2016). Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

[Bottou, 2010] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

[Britz, 2016] Britz, D. (2016). Deep learning for chatbots. `http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/`. Accessed: 2017-10-08.

[Carpenter, 2017] Carpenter, R. (2017). Cleverbot. `http://www.cleverbot.com/`. Accessed: 2017-10-04.

[Cheng et al., 2016] Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

[Chiu et al., 2017] Chiu, C.-C., Lawson, D., Luo, Y., Tucker, G., Swersky, K., Sutskever, I., and Jaitly, N. (2017). An online sequence-to-sequence model for noisy speech recognition. *arXiv preprint arXiv:1706.06428*.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Danescu-Niculescu-Mizil and Lee, 2011] Danescu-Niculescu-Mizil, C. and Lee, L. (2011). Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 76–87. Association for Computational Linguistics.

[Das et al., 2017] Das, A., Kottur, S., Moura, J. M., Lee, S., and Batra, D. (2017). Learning cooperative visual dialog agents with deep reinforcement learning. *arXiv preprint arXiv:1703.06585*.

[Feng et al., 2017] Feng, Y., Zhang, S., Zhang, A., Wang, D., and Abel, A. (2017). Memory-augmented neural machine translation. *arXiv preprint arXiv:1708.02005*.

[Google, 2017] Google (2017). Google assistant. `https://assistant.google.com/`. Accessed: 2017-10-04.

[Havrylov and Titov, 2017] Havrylov, S. and Titov, I. (2017). Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. *arXiv preprint arXiv:1705.11192*.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Jean et al., 2014] Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2014). On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

[Joshi et al., 2017] Joshi, C. K., Mi, F., and Faltings, B. (2017). Personalization in goal-oriented dialog. *arXiv preprint arXiv:1706.07503*.

[Konstas et al., 2017] Konstas, I., Iyer, S., Yatskar, M., Choi, Y., and Zettlemoyer, L. (2017). Neural amr: Sequence-to-sequence models for parsing and generation. *arXiv preprint arXiv:1704.08381*.

[Li et al., 2016a] Li, J., Galley, M., Brockett, C., Spithourakis, G. P., Gao, J., and Dolan, B. (2016a). A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*.

[Li et al., 2016b] Li, X., Mou, L., Yan, R., and Zhang, M. (2016b). Stalematebreaker: A proactive content-introducing approach to automatic human-computer conversation. *arXiv preprint arXiv:1604.04358*.

[Lin et al., 2017] Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

[Lison and Tiedemann, 2016] Lison, P. and Tiedemann, J. (2016). Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles. In *LREC*.

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

[Luong et al., 2014] Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

[Marietto et al., 2013] Marietto, M. d. G. B., de Aguiar, R. V., Barbosa, G. d. O., Botelho, W. T., Pimentel, E., França, R. d. S., and da Silva, V. L. (2013). Artificial intelligence markup language: A brief tutorial. *arXiv preprint arXiv:1307.3091*.

[Microsoft, 2017a] Microsoft (2017a). Cortana. `https://www.microsoft.com/en-us/windows/cortana`. Accessed: 2017-10-04.

[Microsoft, 2017b] Microsoft (2017b). Microsoft bot framework. `https://dev.botframework.com/`. Accessed: 2017-10-04.

[Mitchell, 1998] Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.

[Nallapati et al., 2016] Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

[Olah, 2015] Olah, C. (2015). Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Accessed: 2017-10-08.

[opensubtitles.org, 2017] opensubtitles.org (2017). Opensubtitles. `https://www.opensubtitles.org/`. Accessed: 2017-10-08.

[Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[Sennrich et al., 2015] Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

[Serban et al., 2017] Serban, I. V., Sankar, C., Germain, M., Zhang, S., Lin, Z., Subramanian, S., Kim, T., Pieper, M., Chandar, S., Ke, N. R., et al. (2017). A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*.

[Serban et al., 2016] Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

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

[Shao et al., 2017] Shao, Y., Gouws, S., Britz, D., Goldie, A., Strope, B., and Kurzweil, R. (2017). Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2200–2209.

[Song et al., 2016] Song, Y., Yan, R., Li, X., Zhao, D., and Zhang, M. (2016). Two are better than one: An ensemble of retrieval-and generation-based dialog systems. *arXiv preprint arXiv:1610.07149*.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[Tensorflow, 2017] Tensorflow (2017). Sequence-to-sequence models. `https://www.tensorflow.org/tutorials/seq2seq`. Accessed: 2017-10-08.

[Tiedemann, 2009] Tiedemann, J. (2009). News from opus-a collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, volume 5, pages 237–248.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

[Vinyals and Le, 2015] Vinyals, O. and Le, Q. (2015). A neural conversational model. *arXiv preprint arXiv:1506.05869*.

[Wallace, 2009] Wallace, R. S. (2009). The anatomy of alice. *Parsing the Turing Test*, pages 181–210.

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

[Werbos, 1990] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

[Worswick, 2017] Worswick, S. (2017). Mitsuku. `http://www.mitsuku.com/`. Accessed: 2017-10-04.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

[Xing et al., 2017a] Xing, C., Wu, W., Wu, Y., Liu, J., Huang, Y., Zhou, M., and Ma, W.-Y. (2017a). Topic aware neural response generation. In *AAAI*, pages 3351–3357.

[Xing et al., 2017b] Xing, C., Wu, W., Wu, Y., Zhou, M., Huang, Y., and Ma, W.-Y. (2017b). Hierarchical recurrent attention network for response generation. *arXiv preprint arXiv:1701.07149*.

[Yao et al., 2016] Yao, K., Peng, B., Zweig, G., and Wong, K.-F. (2016). An attentional neural conversation model with improved specificity. *arXiv preprint arXiv:1606.01292*.

[Yu et al., 2017] Yu, Z., Black, A. W., and Rudnicky, A. I. (2017). Learning conversational systems that interleave task and non-task content. *arXiv preprint arXiv:1703.00099*.

[Zhao et al., 2017] Zhao, T., Lu, A., Lee, K., and Eskenazi, M. (2017). Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability. *arXiv preprint arXiv:1706.08476*.

[Zhu et al., 2017] Zhu, Q., Li, Y., and Li, X. (2017). Character sequence-to-sequence model with global attention for universal morphological reinflection. *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 85–89.