A Primer on Seq2Seq Models for Generative Chatbots

VINCENZO SCOTTI, LICIA SBATTELLA, and ROBERTO TEDESCO, DEIB,
Politecnico di Milano, Italy

The recent spread of Deep Learning-based solutions for Artificial Intelligence and the development of Large Language Models has pushed forwards significantly the Natural Language Processing area. The approach has quickly evolved in the last ten years, deeply affecting NLP, from low-level text pre-processing tasks – such as tokenisation or POS tagging– to high-level, complex NLP applications like machine translation and chatbots. This article examines recent trends in the development of open-domain data-driven generative chatbots, focusing on the Seq2Seq architectures. Such architectures are compatible with multiple learning approaches, ranging from supervised to reinforcement and, in the last years, allowed to realise very engaging open-domain chatbots. Not only do these architectures allow to directly output the next turn in a conversation but, to some extent, they also allow to control the style or content of the response. To offer a complete view on the subject, we examine possible architecture implementations as well as training and evaluation approaches. Additionally, we provide information about the openly available corpora to train and evaluate such models and about the current and past chatbot competitions. Finally, we present some insights on possible future directions, given the current research status.

CCS Concepts: • **Computing methodologies** → **Discourse, dialogue and pragmatics**; **Natural language generation**; **Neural Networks**; **Supervised learning;**

Additional Key Words and Phrases: Natural Language Processing, Seq2Seq, Language Model, generative chatbot, Open-domain dialogue

ACM Reference format:
Vincenzo Scotti, Licia Sbattella, and Roberto Tedesco. 2023. A Primer on Seq2Seq Models for Generative Chatbots. *ACM Comput. Surv.* 56, 3, Article 75 (October 2023), 58 pages.
https://doi.org/10.1145/3604281

1 INTRODUCTION

Dialogue agents (also known as conversational agents) have always been a long-running goal of **Artificial Intelligence** (**AI**) since the very beginning of this research field. It is interesting to note that even the well-known Turing test for artificial intelligence was designed around the conversational capabilities of the machine. Since then, dialogue capabilities and *machine intelligence* have been thought as tightly connected.

Nowadays, **Natural Language Processing** (**NLP**), the sub-field of AI focused on human language, offers several approaches to design and implement these agents. Traditionally, NLP literature divides conversational agents into *task-oriented* and *open-domain* [80], as can be seen in the taxonomy depicted in Figure 1. Task-oriented agents (sometimes called *goal-oriented* agents) are

Authors' address: V. Scotti, L. Sbattella, and R. Tedesco, DEIB, Politecnico di Milano, Via Golgi 42, Milano, MI 20133, Italy; emails: {vincenzo.scotti, licia.sbattella, roberto.tedesco}@polimi.it.

0360-0300/2023/10-ART75
https://doi.org/10.1145/3604281

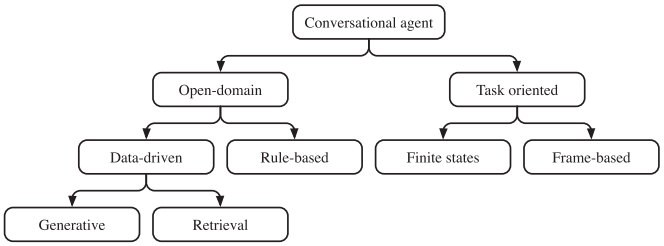


Fig. 1. Conversational agents taxonomy.

thought to be part of an application, acting as an interface, and are further divided into *finite-state* and *frame-based*. Instead, open-domain agents (or *chatbots*) are designed for entertainment and chit-chatting. These agents, in other words, try to give the user the impression of chatting with another human being. Chatbots are divided into *rule-based* and *data-driven*, depending on whether the response is produced following a set of handcrafted rules [191, 196] or following some pattern learnt through statistics or machine learning from dialogic corpora [32, 220]. Finally, data-driven chatbots are further organised into *generative* and *retrieval*. The chatbots from the former group generate the response from scratch, while the latter select the response from a pool of available candidates.

Due to the inherent complex structure of human dialogues, designing and implementing open-domain conversational agents is harder than designing and implementing task-oriented ones, which can rely on the constraints given by the task to accomplish. A popular way to cope with the problem is leveraging machine learning to build the agent, thus delegating to the learning algorithm the duty of identifying the patterns necessary to carry out the conversation. Initially, these chatbots were developed using retrieval approaches due to the high computational power and huge corpora needed to train generative models. In the last years, however, the increased computational power achieved via ***General-Purpose computing on graphics processing units*** (**GPGPU**) enabled the exploitation of **Deep Neural Network** (**DNN**) models to solve many AI problems. With this computational power, training generative models for NLP become possible, yielding very good results where the generated sequences of text were "natural" enough to allow entertaining human-machine conversations [57, 76].

Conversational agents need to deal with sequential data, mainly text. Thus, conversational agents adopt specific Neural Network architectures designed for sequence processing. These architectures for sequence modelling are often called ***Sequence-to-Sequence*** (**Seq2Seq**) since they take sequences as input and produce sequences as output. With the spread of **Deep Learning** (**DL**) techniques in NLP and the availability of large, pre-trained, *Language Models*, Seq2Seq have become the standard architecture to solve many NLP-related tasks, including the development of generative chatbots, as depicted in the example of Figure 2. The application of such architectures, however, is neither limited to generative solutions nor restricted to open-domain conversational agents. In fact, the Seq2Seq architecture is actually compatible with retrieval chatbots or task-oriented agents. In general, Seq2Seq can be seen as a very generic and powerful tool for dealing with a broad range of NLP tasks.

We divide the rest of this article into the following sections. In Section 2 we present the main building blocks from deep learning to create Seq2Seq neural conversational agents. In Section 3 we explain how the aforementioned blocks can be exploited to realise these agents. Finally, in Section 5, we sum up the content of this article and outline the expected future directions and open issues. Additionally, we provide two appendices with material for training and evaluation of Seq2Seq

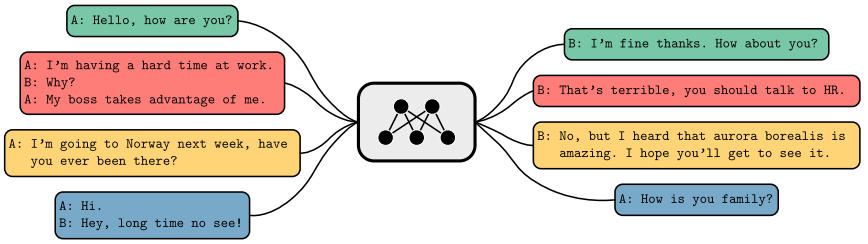


Fig. 2. Generic sequence modelling I/O for chatbots, on the left the *context* and on the right the *generated response*.

neural conversational agents. In Appendix A we present the approaches to train and evaluate the neural Seq2Seq chatbots. In Appendix B we provide details about the available corpora to train chatbots (at different levels of granularity) and the available competitions to test them.

2 BACKGROUND

In this section, we provide preliminary information to understand how DNN models work in general and how Seq2Seq DNNs are built in particular. Additionally, we explain how natural language is encoded into a continuous representation to be used with DNNs and how NLP uses Seq2Seq DNNs.

2.1 Neural Networks for Sequence Modelling

In this section, we introduce the DNN framework applied to sequence processing. We provide an overview on DL and how DNNs work in general. Subsequently, we focus on the two main DNN models for sequence analysis: *recurrent Neural Networks* and *transformer Neural Networks*, the main building blocks of Seq2Seq networks.

2.1.1 Deep Neural Networks. Artificial Neural Networks (or simply Neural Networks) are a flexible and powerful machine learning framework compatible with *supervised*, *unsupervised*, and *reinforcement* learning [62]. They were inspired by the biological neurons of the brain (hence the name). We visualise Artificial Neural Networks in Figure 3.

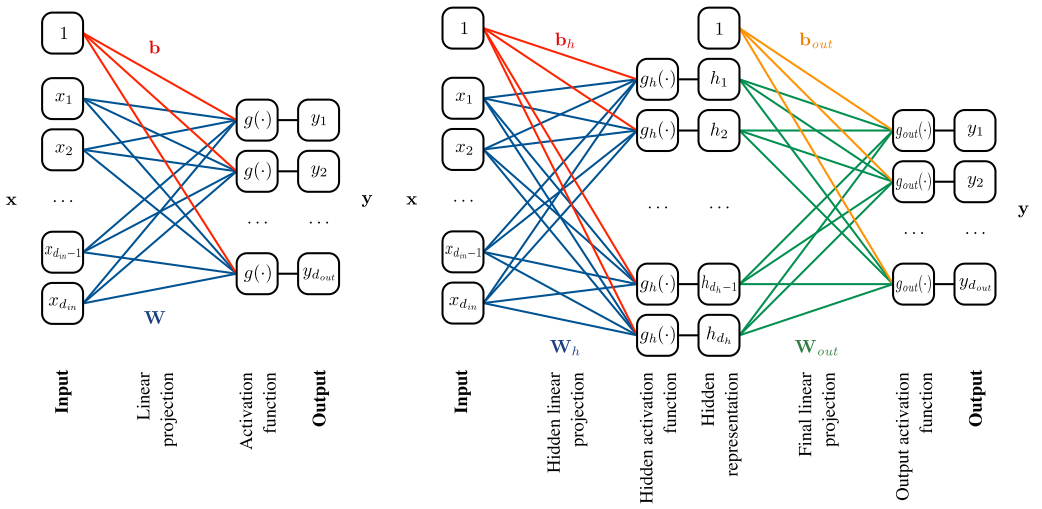
Neurons in the brain are organised into a massive graph. Each is a processing unit that receives electrochemical inputs from other neurons and fires (is triggered) an output signal once the accumulated input reaches a certain threshold. This Neural Network model translates into the mathematical formulation of Equation (1)

$$\mathbf{y} = f(\mathbf{x}; \vartheta) = g(\mathbf{W}^\top \cdot \mathbf{x} + \mathbf{b}), \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^{d_{in}}$ is the input vector, $\mathbf{y} \in \mathbb{R}^{d_{out}}$ is the output vector (with $d_{in} \in \mathbb{N}^+$ and $d_{out} \in \mathbb{N}^+$ being respectively the input and output sizes), $\vartheta \equiv \{\mathbf{W} \in \mathbb{R}^{d_{in} \times d_{out}}, \mathbf{b} \in \mathbb{R}^{d_{out}}\}$ is the set of weights (composed of the matrix $\mathbf{W}$ and the bias vector $\mathbf{b}$), and $g(\cdot)$ is an activation function, like the *sigmoid* $\sigma(\cdot)$ and $\tanh(\cdot)$, or like more complex ones as softmax$(\cdot)$, ReLU$(\cdot)$, and so on.

Initial research on Artificial Neural Networks produced independently the *Perceptron* [108, 146] and the ***ADAptive LINear Element*** (**ADALINE**) [199], two learning frameworks for linear models respectively used for classification problems (i.e., predicting a categorical value from the input) and regression problem (i.e., predicting a continuous value from the input). See Figure 3(a) for further details on the structure of the linear model.

Both models were limited by their linear nature, which could be overcome by adding one or more hidden transformation layers with a non-linear activation inside these models [143]. These



(a) *Artificial Neural Network* (or *perceptron*). (b) *Feed-forward Neural Network* (or *multi-layer perceptron*) with one hidden layer.

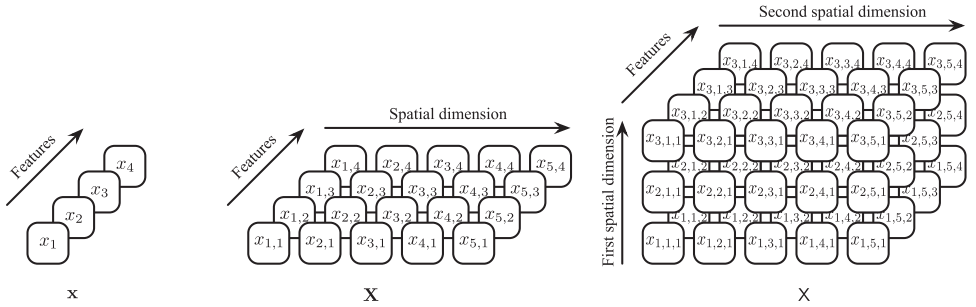
Fig. 3. Artificial Neural Network structure. Connections corresponding to the linear projection are the weights of the networks, the connections departing from the constant value 1 correspond to the bias vector $\mathbf{b}$ and the remaining connection to the weight matrix $\mathbf{W}$. Activation functions are applied element-wise.

multi-layer Neural Networks (see Figure 3(b) for further details on the structure of the multi-layer, non-linear, model), often referred to as *feed-forward Neural Networks*, can ideally approximate (and thus learn) any function, if a sufficient number of neurons in the hidden layer is provided [18]. The output of a feed-forward Neural Network is computed as in Equation (2)

$$\mathbf{y} = f(\mathbf{x}; \vartheta_{out}) = g_{out}(\mathbf{W}_{out}^{\top} \cdot g_h(\mathbf{W}_h^{\top} \cdot \mathbf{x} + \mathbf{b}_h) + \mathbf{b}_{out}) = g_{out}(\mathbf{W}_{out}^{\top} \cdot h(\mathbf{x}) + \mathbf{b}_{out}) = g_{out}(\mathbf{W}_{out}^{\top} \cdot \mathbf{h} + \mathbf{b}_{out}) \quad (2)$$

were $\mathbf{h} \in \mathbb{R}^{d_h}$ is the activation of the hidden layer, with d_h being the number of hidden neurons, $\mathbf{W}_h \in \mathbb{R}^{d_{in} \times d_h}$, $\mathbf{b}_h \in \mathbb{R}^{d_h}$, $\mathbf{W}_{out} \in \mathbb{R}^{d_h \times d_{out}}$, and $\mathbf{b}_{out} \in \mathbb{R}^{d_{out}}$ are, respectively the weights and biases of the hidden and the output layer, and $g_h(\cdot)$ and $g_{out}(\cdot)$ are, respectively, the activations of the hidden and the output layer and $h(\cdot)$ is the hidden transformation function (note that in the formulation we considered a single hidden layer – however, it can be extended to a generic number of hidden layers). In the last ten years, thanks to the increased computation power of GPGPU co-processors, we managed to train feed-forward Neural Networks with a large number of layers, referred to as DNNs. These networks can learn a hierarchy of features in their hidden layers that, provided a sufficient number of training samples, can be transferred and reused for other tasks and in different domains [62].

Neural Networks are parametrised by a set of real-valued weights ϑ. To train these models, it is necessary to define a differentiable *loss function* (or *cost function*, or *objective function*) $\mathcal{L}(\cdot; \vartheta)$ to optimise. $\mathcal{L}(\cdot; \vartheta)$ depends on the parameters of the Neural Network. Common choices for a loss function are the *negative* log-*likelihood* for classification problems and the ***Mean Square Error*** (**MSE**) for regression problems. Training the Neural Network means to iteratively minimise the loss function, updating the weights at each iteration. This iterative optimisation process is usually realised via the *Gradient Descent* algorithm (or some variants like *Stochastic Gradient Descent* and *Mini-Batch Gradient Descent*) [62]. Through the years, many different stochastic optimisations algorithms based on Gradient Descent have been developed [42, 83, 165] to help find the global



(a) One-dimensional tensor, or vector, or feature vector.

(b) Two-dimensional tensor, or matrix, or one-dimensional feature map.

(c) Three-dimensional tensor, or vector, or two dimensional feature map.

Fig. 4. Visualisation of tensors with one, two and three dimensions.

optimum of the loss function corresponding to the best weights combination, avoiding local optima. These stochastic optimisation algorithms leverage the *Backpropagation* algorithm to compute the gradient of the loss function with respect to the weights of the Neural Network.

Due to the complexity and depth of DNNs, many techniques are adopted to regularise and speed up their training process, like *weight regularisation* [18], *dropout* [177], *hidden representation normalisation* [6], and *residual connections* [68]. In fact, despite being a powerful learning tool, many precautions need to be adopted to get the best from these models and not get stuck in local optima during training. Moreover, due to the size of these models, and the huge corpora they usually need, training them from scratch is a very onerous process. For this reason, large models are often trained by big research centres or companies, and publicly released so that they only need to be refined, by means of *fine-tuning*, on a more specific task.

Traditional Artificial Neural Networks, process data samples in form of real values vectors $\mathbf{x} \in \mathbb{R}^d$. Each element of this vector represent an individual feature characterising a sample, this is why we talk of *feature vectors* (see Figure 4(a)). However, these network are a generic tool that can be extended to n-dimensional inputs. We introduce the concepts of n-dimensional *tensor* to indicate generic n-dimensional vector $\mathrm{X} \in \mathbb{R}^{d_1 \times d_2 \times \ldots \times d_n}$ and that of n-dimensional *feature map*, which indicates an $n+1$-dimensional tensor, we visualise these structures in Figure 4. Features maps are collections of feature vectors, organised in a spatial structure. In-fact, some problems allow to exploit the spatial structure of the input in the construction of the DNN. In the case of text and speech, we can see the inputs as sequences of feature vectors, thus as matrices, or two-dimensional tensors, or one-dimensional feature maps (where time is the spatial dimension, see Figure 4(b)). In the case of images, we can see the inputs as a grid of feature vectors, thus as three-dimensional tensors, or two-dimensional feature maps (where the height and width are spatial dimensions, see Figure 4(c)).

Due to language's temporal (i.e., sequential) structure [80], standard Neural Networks are unsuitable for language modelling. In fact, to model human language it is necessary to adopt models capable of capturing *long-term sequential dependencies* in a variable-length context. To this end, *convolutional*, *recurrent* and *transformer* networks are the models of choice as they allow to leverage the spatial structure of the input [80]. Despite convolutional networks being a valid tool for sequence processing, we will focus solely on recurrent and transformer networks as they represent more powerful modelling tools.

In the NLP field, such models have been shown to be powerful tools for building discriminative and generative Seq2Seq Neural Networks for speech and text processing. Recurrent and

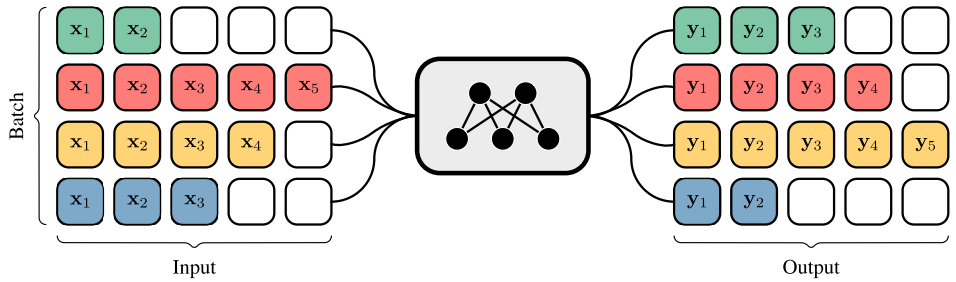


Fig. 5. Visualisation of Seq2Seq model high-level behaviour. The model takes sequences as input and generates sequences as output. It can process batch of sequences. In case of sequences with different lengths, within the same batch, they are usually padded with null values to match the size of the longest one.

transformer networks are usually stacked in a sequence of iterative transformations composing a DNN, and trained on large data collections.

The term Seq2Seq refers to the class of networks built using models that take as input a sequence of vectors

$$X = \langle \mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_{|X|} \rangle, \tag{3}$$

(with $\mathbf{x_t} \in \mathbb{R}^{d_x}$) and generates another sequence of vectors

$$Y = \langle \mathbf{y}_1, \ldots, \mathbf{y}_{t'}, \ldots, \mathbf{y}_{|Y|} \rangle, \tag{4}$$

(with $\mathbf{y_{t'}} \in \mathbb{R}^{d_y}$), as depicted in Figure 5. The hidden transformations of Seq2Seq networks work on sequences of feature vectors encoded into *one-dimensional feature maps*. Such feature maps are actually matrices: $\mathbf{X} \in \mathbb{R}^{d_x \times |X|}$ and $\mathbf{Y} \in \mathbb{R}^{d_y \times |Y|}$ represent the input and output sequences X and Y. The two 1D feature maps are matrices of $|X|$ or $|Y|$ column vectors, where each column vector $\mathbf{x}_t$ or $\mathbf{y}_{t'}$ encodes the tth or t'-th position in the sequence.

$$\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_{|X|}], \tag{5}$$

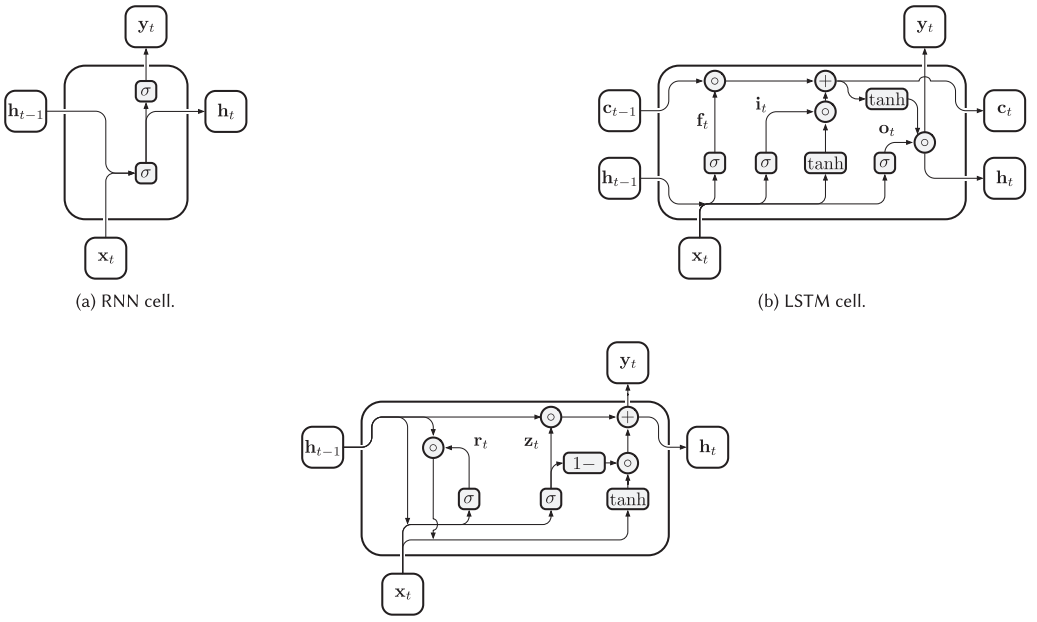
$$\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_{t'}, \ldots, \mathbf{y}_{|Y|}]. \tag{6}$$

Seq2Seq models can handle sequences of undefined length and can process multiple sequences in batch (in this latter case, padding with null values is applied so that all sequences in a batch have the same length[1] – see Figure 5). The overall Seq2Seq network acts as a function, directly yielding the output sequence $\mathbf{Y} = f(\mathbf{X}; \vartheta)$.

2.1.2 Recurrent Neural Networks. Recurrent Neural Networks were the first proposed solution for applying Artificial Neural Networks to a possibly unbound time horizon [80] (see Figure 6). In other words, such networks can be applied to a sequence whose length is neither known a priori not bounded to some maximum value. In a recurrent layer, a cell takes part in a loop where a hidden vector, representing the sequence's past (i.e., the accumulated *memory*), is recurred through the sequence steps (see Figure 7). Interestingly, these networks have been proven to be Turing complete [170].

Several variants of recurrent networks exist: vanilla **Recurrent Neural Networks (RNNs)** [54, 79], **Long Short Term Memories (LSTMs)** [70] and **Gated Recurrent Units (GRUs)** [34]. All these networks can be used to scan a sequence from left to right or vice versa. It is also possible to have bi-directional networks [153] to include information from both sides of the sequence when processing each sequence element (see Figure 8).

[1]This is because, usually, the frameworks to implement DNNs require input and output to be encoded through tensors, and thus, along any given dimension of the tensor, all the elements must have the same length.

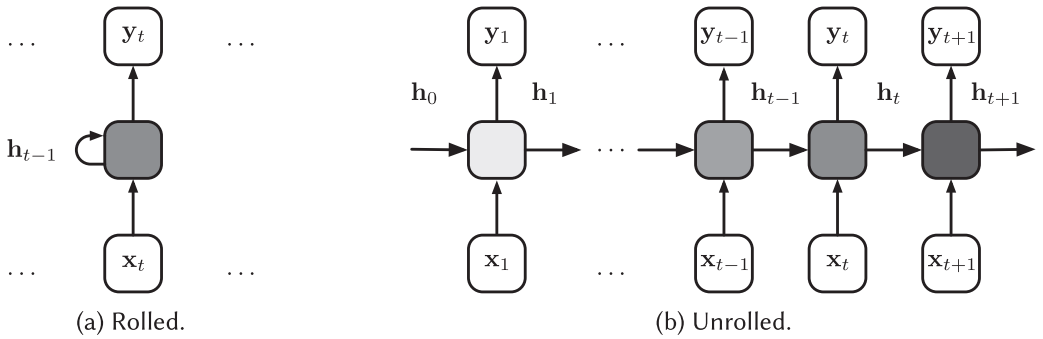


(a) RNN cell.

(b) LSTM cell.

(c) GRU cell.

Fig. 6. Recurrent networks cells. The cell shifts through the elements of the input sequence $\langle \mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_{|X|} \rangle$ to compute the output at each step and compose the resulting sequence $\langle \mathbf{y}_1, \ldots, \mathbf{y}_t, \ldots, \mathbf{y}_{|Y|} \rangle$. At each step, the hidden state vector $\mathbf{h}$ (and the memory vector $\mathbf{c}$, where applicable) is updated and recurred as part of the input for the next sequence step. $\sigma(\cdot)$ is the sigmoid function, $\tanh(\cdot)$ is the hyperbolic-tangent function and $\circ$ is the *Hadamard product* (or element-wise product) operator.

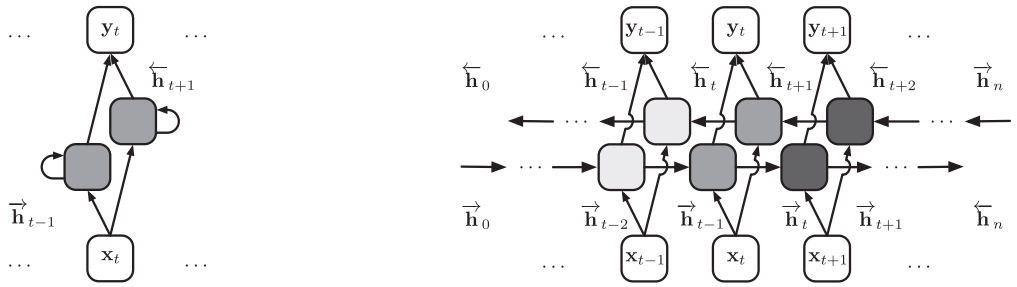


(a) Rolled.

(b) Unrolled.

Fig. 7. Unidirectional RNN sequence processing visualised.

RNN. Vanilla RNNs represented the first important step towards sequence modelling with Neural Networks (see Figure 6(a)). These networks leverage a *hidden memory vector* (also called *hidden context vector*) $\mathbf{h}_t \in \mathbb{R}^{d_h}$ to accumulate past information through the sequence. For each element $\mathbf{x}_t \in \mathbb{R}^{d_x}$ of the input sequence and the corresponding hidden memory vector from the previous step $\mathbf{h}_{t-1}$, the RNN updates such hidden memory vector and computes the output $\mathbf{y}_t \in \mathbb{R}^{d_y}$ for the current time step t.



(a) Rolled. (b) Unrolled.

Fig. 8. Bi-directional RNN sequence processing visualised.

Elman's original formulation [54] prescribed to compute the hidden memory vector $\mathbf{h}_t$, given $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$, as reported in Equation (7) and then calculate $\mathbf{y}_t$ as reported in Equation (8). Matrices $\mathbf{W}_x \in \mathbb{R}^{d_x \times d_h}$ and $\mathbf{W}_h \in \mathbb{R}^{d_h \times d_h}$, and vectors $\mathbf{b}_h \in \mathbb{R}^{d_h}$ are the weights of the hidden layer, matrix $\mathbf{W}_y \in \mathbb{R}^{d_h \times d_y}$ and vector $\mathbf{b}_y \in \mathbb{R}^y$ are the weights of the output layer, and $\sigma(\cdot)$ is the sigmoid function:

$$\mathbf{h}_t = \sigma\left(\mathbf{W}_x^\top \cdot \mathbf{x}_t + \mathbf{W}_h^\top \cdot \mathbf{h}_{t-1} + \mathbf{b}_h\right), \quad (7) \qquad \mathbf{y}_t = \sigma\left(\mathbf{W}_y^\top \cdot \mathbf{h}_t + \mathbf{b}_y\right). \quad (8)$$

Instead, in Jordan's RNNs, $\mathbf{h}_t$ is updated using the $\mathbf{y}_{t-1}$ instead of $\mathbf{h}_{t-1}$, as reported in Equation (9):

$$\mathbf{h}_t = \sigma\left(\mathbf{W}_x^\top \cdot \mathbf{x}_t + \mathbf{W}_h^\top \cdot \mathbf{y}_{t-1} + \mathbf{b}_h\right). \quad (9)$$

RNNs are trained by means of the *backpropagation through time* approach, which implies calculating the derivatives throughout the input sequence and computing the product of all these values. The multiplications of the partial derivatives, yield by the chain rule, to compute the overall gradient of the error could lead to the *explosion* or the *vanishing* of the norm of such gradient (making it impossible to train the network) [125]. As a consequence, vanilla RNNs are not suitable for tackling long-term dependencies, which are common in natural language.

LSTM. LSTMs were designed to overcome some limitations of the original RNNs [70]. Despite the sound mathematical formulation behind RNNs, training those networks turns out to be a hard task [125]. This is due to their lack of robustness to noise and the aforementioned vanishing and exploding gradient problems.

LSTMs solved the noise robustness and vanishing gradient issues, enabling the modelling of longer sequences, through *gating mechanisms*. The exploding gradient issue may still occur even in these networks. However, the use of techniques like *gradient clipping* [125] can help to cope with this issue.

At each sequence step, these LSTM networks take as input $\mathbf{x}_t \in \mathbb{R}^{d_x}$, yield an *output hidden vector* $\mathbf{h}_t \in \mathbb{R}^{d_h}$ –which is passed to and updated by each step of the sequence analysis– and maintain an internal *cell state vector* (or memory vector) $\mathbf{c}_t \in \mathbb{R}^{d_h}$, which is updated at each step and represents the current status of the network. This network cell uses three gates to model the time sequence: the *input gate* $\mathbf{i}_t \in \mathbb{R}^{d_h}$ controls the amount of new information set in the cell state (see Equation (10)), the *forget gate* $\mathbf{f}_t \in \mathbb{R}^{d_h}$ controls the amount of information discarded from the cell state (see Equation (11)), and, finally, the *output gate* $\mathbf{o}_t \in \mathbb{R}^{d_h}$ controls the amount of information set in the output (see Equation (12)). Given these gates, $\mathbf{h}_t$ and $\mathbf{c}_t$ are then updated according to Equations (13) and (14), respectively.

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_{xi}^\top \cdot \mathbf{x}_t + \mathbf{W}_{hi}^\top \cdot \mathbf{h}_{t-1} + \mathbf{b}_i\right), \quad (10)$$

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_{xf}^\top \cdot \mathbf{x}_t + \mathbf{W}_{hf}^\top \cdot \mathbf{h}_{t-1} + \mathbf{b}_f\right), \quad (11)$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_{xo}^\top \cdot \mathbf{x}_t + \mathbf{W}_{ho}^\top \cdot \mathbf{h}_{t-1} + \mathbf{b}_o\right). \quad (12)$$

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh\left(\mathbf{W}_{xc}^\top \cdot \mathbf{x}_t + \mathbf{W}_{hc}^\top \cdot \mathbf{h}_{t-1} + \mathbf{b}_c\right). \quad (13)$$

$$\mathbf{h}_t = \mathbf{y}_t = \mathbf{o}_t \circ \tanh\left(\mathbf{c}_t\right). \quad (14)$$

In Equations (10) to (14) the matrices $\mathbf{W}_{xi}, \mathbf{W}_{xo}, \mathbf{W}_{xf} \in \mathbb{R}^{d_x \times d_h}$ and $\mathbf{W}_{hi}, \mathbf{W}_{ho}, \mathbf{W}_{hf} \in \mathbb{R}^{d_h \times d_h}$, and the vectors $\mathbf{b}_i, \mathbf{b}_o, \mathbf{b}_f \in \mathbb{R}^{d_h}$ are the parameters of the LSTM cells, and symbol $\circ$ is the *Hadamard product* (or element-wise product) operator. Figure 6(b) shows the typical structure on an LSTM cell.

GRU. GRUs, like LSTMs, leverage a gating mechanism to control memory [34], but reduce such gates to two. GRUs maintain the same computational capabilities as LSTMs and, on small corpora, they show better results [35].

The gates of a GRU cell are called *update* and *reset*. The update gate is a combination of the input and output gate of LSTMs and computes $\mathbf{z}_t \in \mathbb{R}^{d_h}$ as shown in Equation (15). The reset gate plays the same role as the LSTM forget gate and calculates $\mathbf{r}_t \in \mathbb{R}^{d_h}$ as shown in Equation (16). Unlike LSTMs cells, GRUs use a single hidden vector $\mathbf{h}_t \in \mathbb{R}^{d_h}$ to represent both memory and output. Finally, the output $\mathbf{y}_t = \mathbf{h}_t$, at each step, is computed as in Equation (17). Figure 6(c) shows the typical structure of a GRU cell.

$$\mathbf{z}_t = \sigma\left(\mathbf{W}_{xz}^\top \cdot \mathbf{x}_t + \mathbf{W}_{hz}^\top \cdot \mathbf{h}_{t-1} + \mathbf{b}_z\right), \quad (15) \qquad \mathbf{r}_t = \sigma\left(\mathbf{W}_{xr}^\top \cdot \mathbf{x}_t + \mathbf{W}_{hr}^\top \cdot \mathbf{h}_{t-1} + \mathbf{b}_r\right), \quad (16)$$

$$\mathbf{y}_t = \mathbf{h}_t = (1 - \mathbf{z}_t) \circ \mathbf{h}_{t-1} + \mathbf{z}_t \circ \tanh\left(\mathbf{W}_{xh}^\top \cdot \mathbf{x}_t + \mathbf{W}_{rh}^\top \cdot (\mathbf{r}_t \circ \mathbf{h}_{t-1}) + \mathbf{b}_h\right). \quad (17)$$

Bi-directional. Bi-directionality was introduced to improve the modelling capabilities of RNNs [153]. This approach is applicable to any of the aforementioned recurrent cells.

Usually, RNNs used in NLP apply only a *forward* analysis, as shown in Figure 7(b). Thus, each hidden state holds the information coming from the preceding positions. The bi-directional approach prescribes scanning the sequence *forwards* and *backwards*, and extracting distinct hidden vectors per direction $\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$, as depicted in Figure 8(b). Then, $\overrightarrow{\mathbf{h}}_t$ and $\overleftarrow{\mathbf{h}}_t$ can be combined in many different ways (e.g., sum or concatenation). It is also possible to learn a separate point-wise transformation $\mathbf{W}_h$ to combine the two vectors, as in the example of Equation (18), where $f_{combine}(\cdot)$ can be any operation combining two vectors (e.g., concatenation, sum):

$$\mathbf{h}_t = \mathbf{W}_h^\top \cdot \left(f_{combine}\left(\overrightarrow{\mathbf{h}}_t, \overleftarrow{\mathbf{h}}_t\right)\right). \quad (18)$$

As can be inferred from Equation (18), the improved modelling capabilities given by the bi-directional approach come at the cost of increased memory requirements. In fact, while uni-directional models only need to store and update a single memory vector, the bidirectional approach requires keeping in memory the entire sequence of hidden memory states for both directions. However, this requirement does not hold when the last hidden state is the only information the task needs, for example when doing a sequence summary.

2.1.3 Transformer Neural Networks. Transformer networks arose from the need to overcome the so-called *sequential limitations* of RNNs. In fact, RNNs necessarily require a *sequential input processing* and thus operations cannot be parallelised across the sequence: RNNs need to process $\mathbf{x}_t$ *before* moving to $\mathbf{x}_{t+1}$. Instead, Transformer modules rely on *highly parallelisable transformations*. Moreover, Transformers yielded better results than RNNs because of their improved (i.e., longer) context management capabilities [46, 134].

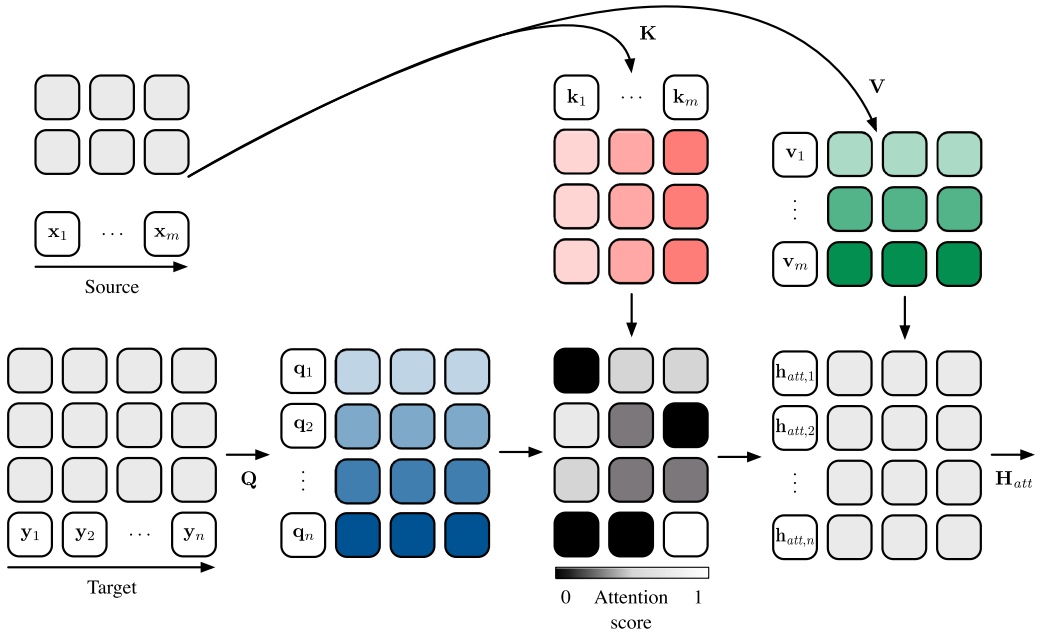


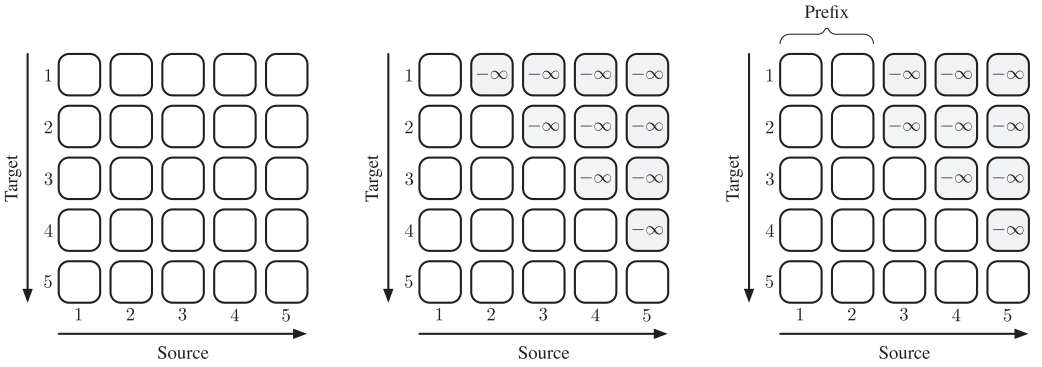
Fig. 9. Attention mechanism processing a sequence. Grey-scaled boxes represent the attention score between the input and output element. Brighter cells of the attention matrix correspond to higher attention scores and vice versa. The attention mechanism learns the alignment between the input sequence (projected into keys $\mathbf{K}$) and output sequence (projected into queries $\mathbf{Q}$). This alignment is used to combine the elements of the input sequence (projected into values $\mathbf{V}$) and produce a new hidden representation of the output sequence $\mathbf{H}_{att}$. Notice that the combination is repeated for each element of the output sequence, yielding an hidden representation of the same length.

Transformer architectures revolve around the *attention mechanism* [7]. This mechanism was initially developed for RNNs as a methodology to find the correspondences between the elements of input and output sequences, but eventually, it completely substituted the sequence analysis modules [101, 186]. The original application was in machine translation [186], were the effect of the learnt alignment is to find connections as in Figure 13.

Attention mechanism. The attention mechanism allows a network to extract and use information from arbitrarily large contexts [80]. Attention layers learn to weight a combination of feature vectors, depending on some query.

In its generic formulation, the attention mechanism takes two 1D feature maps of variable length as input: the *source* $\mathbf{X} \in \mathbb{R}^{d_x \times m}$ and the *target* $\mathbf{Y} \in \mathbb{R}^{d_y \times n}$. The target $\mathbf{Y}$ is used to generate the *queries* $\mathbf{Q} \in \mathbb{R}^{d_k \times n}$ (i.e., $\mathbf{Y}$ describes the *required* information), while $\mathbf{X}$ is used to provide *keys* $\mathbf{K} \in \mathbb{R}^{d_k \times m}$ and *values* $\mathbf{V} \in \mathbb{R}^{d_v \times m}$, which are calculated through linear projections as $\mathbf{Q} = \mathbf{W}_q^\top \cdot \mathbf{Y}$, $\mathbf{K} = \mathbf{W}_k^\top \cdot \mathbf{X}$, and $\mathbf{V} = \mathbf{W}_v^\top \cdot \mathbf{X}$. Such matrices $\mathbf{W}_q \in \mathbb{R}^{d_y \times d_k}, \mathbf{W}_k \in \mathbb{R}^{d_x \times d_k}, \mathbf{W}_v \in \mathbb{R}^{d_x \times d_v}$ contain the trainable parameters of the attention mechanism. All operations are illustrated in Figure 9.

The overall attention mechanism and the computation of the attention function $f_{attention}(\cdot)$ is described in Equation (19) and depicted in Figure 11(a). Keys $\mathbf{K}$ are matched against the queries $\mathbf{Q}$ through a *scoring function* (the scoring function is the *cross-correlation*, computed as $\mathbf{Q}^\top \cdot \mathbf{K}$). These scores are then scaled (by a $1/\sqrt{d_k}$ factor, to have unit variance of the scores) and normalised row-wise through a softmax$(\cdot)$ function (in this way the resulting values will sum up to 1). The



(a) Fully visible. (b) Causal. (c) Prefix.

Fig. 10. Examples of self-attention masking patterns with $t = 5$. Blank positions correspond to 0 values. The mask is summed to the un-normalised attention scores, before the softmax(·) activation.

computed values in the resulting matrix are used as *weights* in the linear combination of values $\mathbf{V}$. They represent how much information to retain from the value $\mathbf{v}_j$ (the jth row of $\mathbf{V}$), corresponding to the key $\mathbf{k}_j$, when computing the combination for query $\mathbf{q}_i$:

$$\mathbf{H}_{att}^{\top} = f_{attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}^{\top} \cdot \mathbf{K}}{\sqrt{d_k}}\right) \cdot \mathbf{V}^{\top}, \tag{19}$$

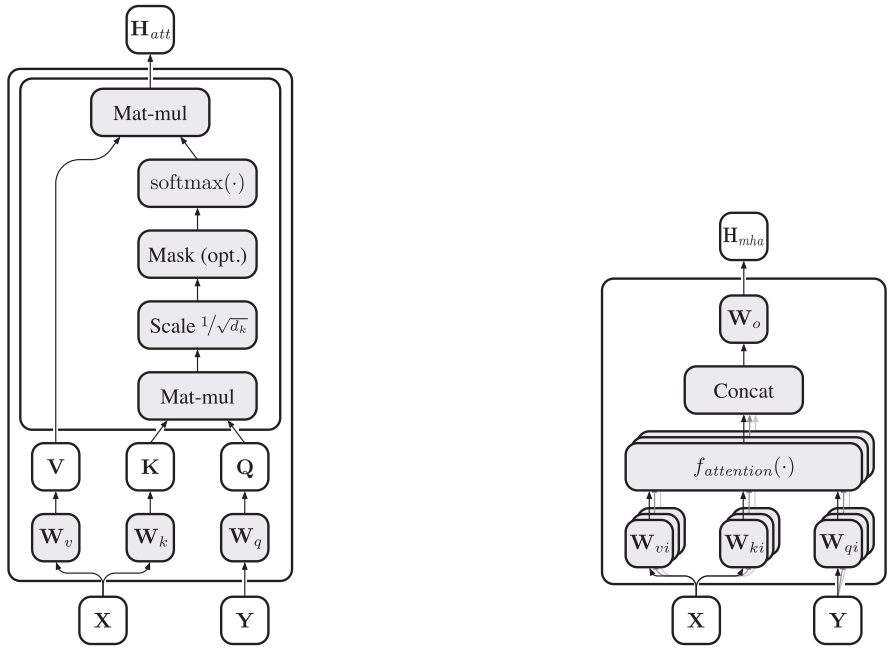
where $\mathbf{H}_{att} \in \mathbb{R}^{d_v \times n}$ is the output feature map composed of the output hidden vectors, each corresponding to a position of the target feature map.

To explain this approach, consider the classic language translation task, where X and Y represent the sentence to be translated and a candidate translation. We want to know how strong the relationship between any word in the sentence and any word of the candidate translation is (in this way, we provide the rest of the translation network with further information that can be useful for selecting the best translation). Thus, we compute Q and K, which are a way of projecting X and Y to a common space where they can be matched against each other. The resulting matching weights are applied to V, which is still X but projected on the space we need to obtain as output.

The attention mechanism presented so far is the generic *cross-attention*. Transformer networks mostly use the *self-attention variant*, where the source and target are the same: $\mathbf{X} = \mathbf{Y}$. Self-attention relates different positions of a single sequence $\mathbf{X}$ in order to compute a more effective representation of $\mathbf{X}$. Moreover, self-attention allows for *masking patterns* in the computation of attention.

A masking pattern is represented by a matrix $\mathbf{M}$ whose values are either 0 or $-\infty$ (see Figure 10) and is summed to the matrix of key-query scores: $\mathbf{Q}^{\top} \cdot \mathbf{K} + \mathbf{M}$. If all the elements of $\mathbf{M}$ are 0, the sequence is *fully visible*, otherwise, $\mathbf{M}$ is used to prevent the attention from considering some specific positions. For example, if an element $m_{i,j} = -\infty$, the weight corresponding to that position, after the softmax(·) normalisation becomes 0 (i.e., no information from the corresponding value is retained). This fully-visible attention is typical of bi-directional encoder models, the encoder part of a transducer model, and the cross-attention of transducers as well, as we will explain in Sections 2.2.2 and 3.1.

If the upper right triangular sub-matrix is set to $-\infty$, the mask forces *casualty* (i.e., sequentiality) in the computation of the output hidden representation. This covered attention is typical of causal decoders models, and the decoders parts of transducers, as we explain in Sections 2.2.2 and 3.1.



(a) Attention mechanism. The inner box is the computation scheme of $f_{attention}(\cdot)$

(b) Multi-head attention mechanism.

Fig. 11. Attention mechanism computation visualisation.

Finally, a *prefix* mask that is causal only for the right portion of the mask, and it is fully visible in the left portion. This pattern permits to have fully visible attention to the initial part of the input and to analyse autoregressively the remaining part. The partially covered attention, which is similar to a combination of the patterns used by transducers, is rarely used and allows building a *non-causal decoder* [193]: a single model that is a mixture of a bi-directional encoder and a causal decoder without the intermediate cross-attention [9, 137].

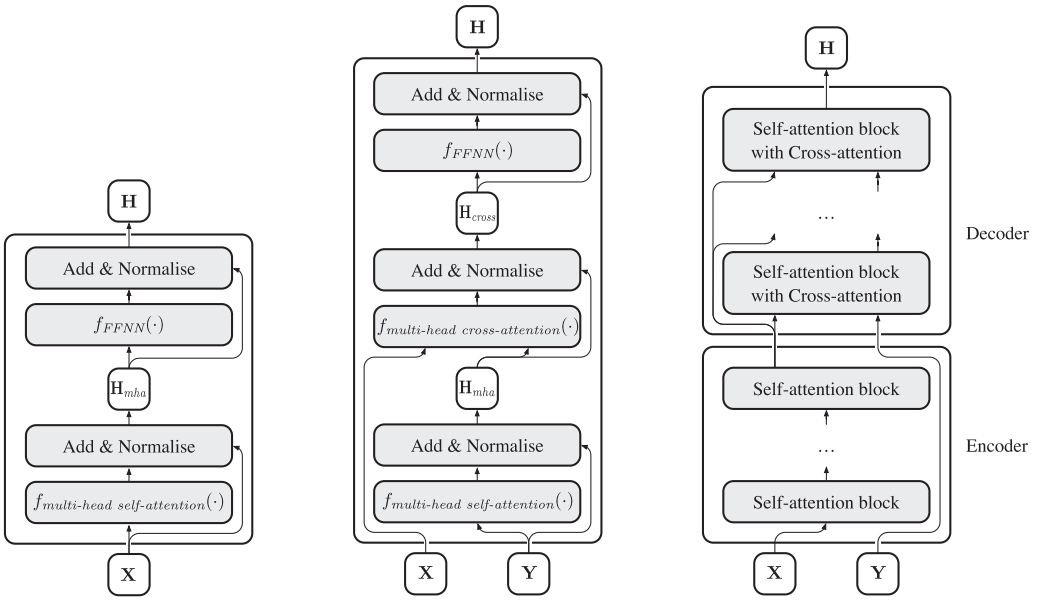
Masking is also used to deal with sequences of varying lengths when doing batch processing. In fact, since sequences in a batch are required to have the same length, they need to be *padded* to reach the required size: the columns and rows of $\mathbf{M}$ corresponding to padded positions are set to $-\infty$. This approach holds for both self- and cross-attention.

To capture the different parallel relationships that could occur within a sequence, the attention is often parallelised through *multiple attention heads* (see Figure 11(b)). Each of the h heads computes the attention transformation with its weights ($\mathbf{W}_{q,i}$, $\mathbf{W}_{k,i}$, $\mathbf{W}_{v,i}$, with $i \in [1, h] \subseteq \mathbb{N}$), independently of the others. The outputs are then concatenated along the feature dimension and merged through a further linear transformation, as described by Equation (20):

$$\mathbf{H}_{mha}^{\top} = f_{multi\text{-}head\ attention}(\mathbf{X}, \mathbf{Y}) = \mathbf{W}_o^{\top} \cdot (\mathbf{H}_{att,1} \oplus \ldots \oplus \mathbf{H}_{att,h}), \tag{20}$$

where $\mathbf{H}_{att,i} \in \mathbb{R}^{d_v \times n}$ is the output of the ith head, $\mathbf{W}_o \in \mathbb{R}^{(d_v \cdot h) \times d_{mha}}$ is a matrix of trainable parameters, $\oplus$ represents the concatenation operator (applied along the feature axis), and $\mathbf{H}_{mha} \in \mathbb{R}^{d_{mha} \times n}$ is the resulting hidden representation.

Transformer blocks. Transformer networks (see Figure 12(c)) build their hidden representations by stacking *transformer blocks* [186] (see Figure 12(a) and 12(b)). Blocks alternate a multi-head attention sub-block (yielding an intermediate representation $\mathbf{H}_{mha} \in \mathbb{R}^{d_{mha} \times n}$) with a ***Feed-Forward***



(a) Self-attention block.

(b) Self-attention block with cross-attention.

(c) Full transformer network

Fig. 12. Transformer blocks and network visualisation.

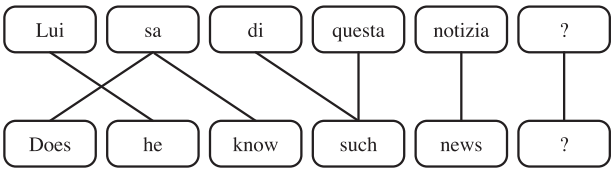


Fig. 13. Translation alignment between the Italian sentence "Lui sa di questa notizia?" and its English translation "Does he know such news?". The mapping is quite complex, as it is neither monotonic (can go forward and then backwards) nor one-to-one (single words in one language correspond to multiple in the other and vice-versa).

Neural Network (**FFNN**) sub-block (yielding an output representation $\mathbf{H} \in \mathbb{R}^{d_h \times n}$). Usually, each sub-block adds a residual connection around the transformation and a layer-normalisation step [6] right after the sum of the residual connection (in some cases, the order of the operations is different [135]).

The FFNN is a *point-wise non-linear transformation*. As can be seen in Equation (21), this transformation is computed as a sequence of linear projections with a non-linear activation function in the middle:

$$\mathbf{h}_i = f_{FFNN}(\mathbf{h}_{mha,i}) = \mathbf{W}_{out}^{\top} \cdot \text{ReLU}\left(\mathbf{W}_{in}^{\top} \cdot \mathbf{h}_{mha,i} + \mathbf{b}_{in}\right) + \mathbf{b}_{out}, \tag{21}$$

where $\mathbf{h}_{mha,i} \in \mathbb{R}^{d_h}$ is a single hidden vector corresponding to the i-th position in the input sequence, while matrices $\mathbf{W}_{in} \in \mathbb{R}^{d_{mha} \times d_{ffnn}}$, $\mathbf{W}_{out} \in \mathbb{R}^{d_{ffnn} \times d_h}$ and vectors $\mathbf{b}_{in} \in \mathbb{R}^{d_{ffnn}}$, $\mathbf{b}_{out} \in \mathbb{R}^{d_h}$ contain the trainable parameters of $f_{FFNN}(\cdot)$. The non-linear activation function does not need to be a ReLU$(\cdot)$; GELU$(\cdot)$ is used in some implementations [46, 134].

The main point of transformer blocks is that all the transformations in the same layer can be computed in parallel. Thus the process is not slowed down by the sequential analysis. The

drawback is in the increased memory requirement. However, software and hardware techniques exist to reduce space complexity (and also improve the temporal complexity) [110, 110, 133].

In the case of transducer architectures with encoder and decoder (more on this in Section 3.1) there is also an additional cross-attention sub-block between the self-attention and the FFNN sub-blocks. The output from the first self-attention transformation $\mathbf{H}_{mha}$ is used as the target in a further attention transformation sub-block. However, this sub-block is a cross-attention sub-block and not a self-attention one: a separate source $\mathbf{X}$ is used to compute keys and values. This cross-attention transformation is done to find alignment between input and output sequences. Usually, as depicted in Figure 12(c), the input sequence is pre-processed through an encoder and the output sequence through a decoder (this *transducer* approach is not mandatory, however, as we explain in Section 2.2.2).

2.2 Text Representation and Processing

In this section, we introduce how Neural Networks represent and manipulate text; in particular, methods to map text from a *discrete orthogonal* space to a *dense compact* space, and probabilistic models applied to language.

2.2.1 Vector Semantics and Embeddings. Before going into the details of *vector semantics*, we start with some definitions. We call *vocabulary* $\mathcal{V}$ a set of character sequences, a *word type* (or *word*) w is a unique entry in a vocabulary, while a *token* represents a word instance in some text. Often, words represent *flexed forms* of the same base form, which is called *lemma* (e.g., words being, been, is are flexed form of the lemma to be). Embedding techniques can be applied to words, tokens or lemmas, to transform them into continuous-values vectors: the *embeddings*.

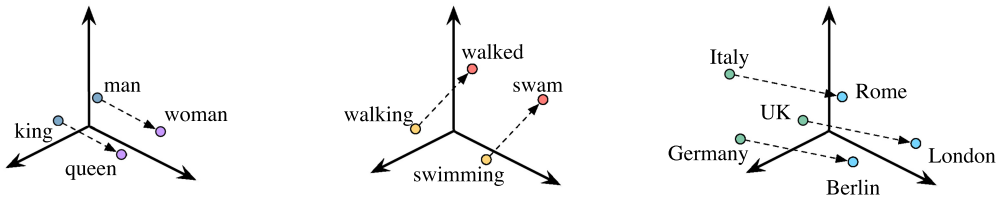
Due to the vocabulary size, Neural Network models tend to grow large in the number of parameters. However, an advantage we shall see of Seq2Seq models is that, since they build and transform their embedding representation from an entire text sequence –rather than single words– they can leverage *sub-word tokenisation* to encode the input text reducing the number of symbols (and thus the parameters to embed such symbols). With this sub-word approach, the vocabulary contains sequences of frequent sub-words. For example, tokenisation can be represented as token and isation. With this approach, words are decomposed into smaller units (down to single character level) that are the actual constituents of the vocabulary, allowing also to manage *out-of-vocabulary words*. Usually, these sub-words units are extracted from data applying dictionary-based *compression algorithms* like ***Byte-Pair Encoding*** (**BPE**) [156].

Human language is encoded by means of orthogonal symbols (alphabetic characters, ideograms, diacritics, etc.), which form sequences that group at various granularity levels. We call such groups *words*, *sentences*, *sections*, and so on. Several *discrete* representations exist to encode such sequences. For example, the popular *one-hot encoding* transforms a word into a vector $\mathbf{o} \in \mathbb{1}^{|\mathcal{V}|}$, such that $\|\mathbf{o}\|_2 = 1$ In particular, all the elements are zero except the one corresponding to the word to be encoded, which is set to 1. Note that $\mathcal{V}$ is usually a large set (with millions of elements).

$$\mathbf{o} = [0, \ldots, 0, 1, 0, \ldots, 0]. \tag{22}$$

Deep learning models based on Neural Networks, however, work better on *dense* representations expressed as *tensors*,[2] and referred to as *vector semantics*. In fact, through vector semantics, it is possible to project human language symbols and sequences into dense, smooth and compressed representations. Thus, the key idea of deep learning models for NLP is to project everything into

[2]Here, with *tensor* we mean a generalised, multidimensional array.



(a) Female-male. (b) Verbe tense. (c) Country-capital.

Fig. 14. Examples of semantic space properties. Points in the space represent the vectors of the associated words, and the relative position of the points corresponds to certain relations.

a continuous d-dimensional space (where $d \ll |\mathcal{V}|$) and then manipulate such representation. For example, a sequence of tokens X_{sparse} is converted into a sequence of vectors X_{dense}:

$$X_{sparse} = \langle x_1, \ldots, x_t, \ldots, x_{|X|} \rangle \rightarrow X_{dense} = \langle \mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_{|X|} \rangle, \tag{23}$$

where $|X_{sparse}| = |X_{dense}| = |X|$, $x_t \in \mathcal{V}$ and $\mathbf{x}_t \in \mathbb{R}^d$. This sequence can be further converted into a matrix $\mathbf{X}$ or a tensor to be processed by a Seq2Seq model.

$$X_{dense} = \langle \mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_{|X|} \rangle \rightarrow \mathbf{X} \in \mathbb{R}^{|X| \times d}. \tag{24}$$

A crucial property characterises vectors in this space: they represent the semantic (and, sometimes, syntactic) meaning of the pieces of text they encode [80]. Figure 14 shows some examples about the encoding of words. Thus, it is possible to compute the *semantic similarity* among pieces of text by computing the *distance* of their corresponding vectors. These semantic vector representations are called *embeddings*, but are in practice feature vectors.

In the last years, various approaches emerged to encode *word embeddings*, using them as a basic "building block" for models representing more complex, higher-level structures, such as sentences, sections and even whole documents.

Word embeddings. As introduced above, models for word embeddings encode words into a semantic space, where they are represented as d-dimensional vectors. These models can be grouped according to two orthogonal criteria: *count-based* vs. *prediction-based* models, and *shallow* (and thus static) vs. *deep* (and thus contextual) models.

Shallow models represent the oldest embedding approach [36–38, 43, 44, 52]. They are encoded in an embedding matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where $\mathcal{V}$ is the vocabulary and d is the desired embedding space dimension. The target word's sparse (one-hot) representation $\mathbf{o}$ is used to fetch the word embedding $\mathbf{u} \in \mathbb{R}^d$ from the embedding matrix $\mathbf{W}$, as: $\mathbf{u} = \mathbf{W}^\top \cdot \mathbf{o}$. Notice that the one-hot encoding and the multiplication shown in the previous equation are actually implemented by fetching the word embedding from the matrix (i.e., a row) starting from the index of the target word.

In particular, prediction-based models are trained to predict a target word, given a context window of surrounding words in the corpus samples (Continuous Bag-of-Word - CBoW approach) or to predict the surrounding context words given the target word (skip-gram approach); examples are Word2Vec [112, 113] and fastText [19]. Instead, count-based models are trained using word co-occurrence counts in the corpus [12]; see, for example, GloVe [126].

Deep contextual models have been around for some time [13]. However, they gained traction recently, due to the availability of sufficient computational power to train them on large corpora, in a reasonable amount of time. The idea behind such models is to leverage all the elements in the input word sequence to build a sequence of *hidden*, compact, vector representations useful to

predict the next unknown word (or generic missing words). Hidden representations extracted by these models encapsulate information on both the corresponding input token and all the other tokens of the sequence. Due to this property, we talk of contextual/contextualised embeddings: the entire sequence serves as context to encode all tokens, and this is what gives deep models an advantage over shallow ones.

Contextual models are based on DNNs and, since they are trained to predict the word sequence probability distribution, represent a typology of (probabilistic) **Language Models** (**LMs**). Thus they fall into the group of predictive models. We refer to Section 2.2.2 for further details on probabilistic language models.

Early deep contextual models were implemented using unidirectional recurrent Neural Networks [14, 16]. *ELMo* [127], instead, was the first example of bi-directional recurrent networks applied for this problem. Nowadays, these models are built using state-of-the-art transformer networks [101], *GPT* [134] and *BERT* [46] are examples of transformer based language models. Note that, independently of the implementation of the hidden layers, all deep models start from an initial shallow embedding of each word in the sequence. The goal of the hidden layers is thus to refine these initial vectors, generating better, more semantically informative embeddings by incorporating information from the other tokens in the input (context) sequence.

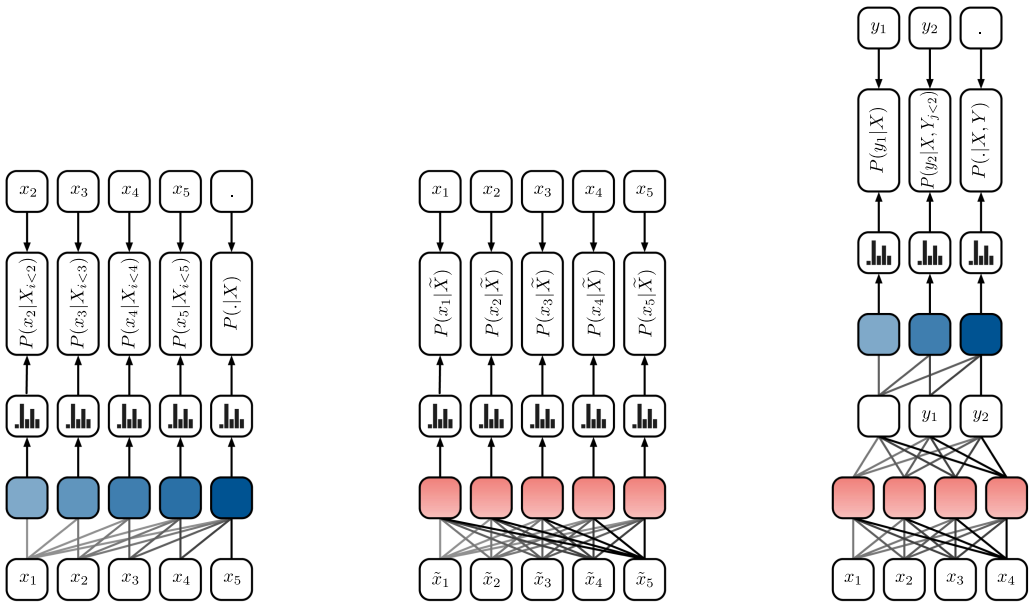
Generalised embeddings. Besides word-level embeddings, other embeddings are employed in NLP. These generalised embeddings try to encode information of longer pieces of text (e.g., sentences, paragraphs, documents, ...) into single vectors. Although deep contextual approaches for word-level embeddings represent the most adopted solution (due to their performances), generalised embeddings still represent a useful tool, as they are simple, fast and –for several NLP tasks– provide good-enough embeddings.

Sentence embeddings represent the most adopted typology of generalised embeddings. They find applications in many fields, like document retrieval, and allow for very compact meaningful representations. Sentence embeddings are divided into two groups: *parametrised* and *non-parametrised* models. Parametrised models must be trained either through supervised approaches –leveraging corpora for **Semantic Textual Similarity** (**STS**)[3] or **Natural Language Inference** (**NLI**)[4] tasks [142]– or unsupervised approaches, leveraging generic corpora for language modelling [85, 121]. Instead, non-parametrised models are built on top of word-level embedding models, and thus training is not required [5, 219].

Parametrised models are similar to word embeddings, and can be either shallow or built on top of deep language models. To train a supervised model of this kind, a labelled corpus on STS or NLI is needed. Sentence-BERT [142] is a popular example of these models. Instead, it is sufficient to leverage a generic, unlabelled corpus to train an unsupervised sentence embedding model. Models like Sent2Vec [121] and Skip-Thought [85], leveraging a self-supervised approach, are examples of model that can leverage unlabelled corpora. They are trained to predict the missing words in a sentence or the following sentence (word by word) in a sequence, respectively.

Non-parametrised models showed that it is possible to achieve meaningful representations simply by combining existing word embeddings. Models like SIF [5] or DynaMax [219] build their sentence representation starting from the sequence of word embeddings constituting the sentence to encode, and then applying a *weighted average pooling* layer or a *max pooling* layer, respectively. Although non-parametrised models do not achieve the results of parametrised ones, they are easy to implement and require little computational resources.

[3]Models for STS are used to compute a score about the semantic similarity of two sentences.
[4]Models for NLI compare two sentences to understand if they are *consequential*, *contradictory* or *independent*.



(a) Causal LM. (b) Bi-directional LM. (c) Transducer LM

Fig. 15. Seq2Seq Neural Network architectures for different language modelling approaches. $X \in \mathcal{V}^{|X|}$ and $Y \in \mathcal{V}^{|Y|}$ are sequences of tokens. The . symbol is the stopping symbol that indicates the end of a sequence. Red and blue boxes correspond to hidden representations computed by the Seq2Seq . Connections correspond to the dependencies in the hidden representation computation.

Apart from sentence embeddings, other high-level embedding models include documents, *knowledge graphs*,[5] and even *speaker persona*[6] in conversations. These can be employed in many NLP applications, like conversational agents.

2.2.2 Probabilistic Language Models. *Probabilistic language models*, or simply LM, are *probability distributions over sequences of words* $P_{LM}(w_1, \ldots, w_i, \ldots, w_n)$ (with $w_i \in \mathcal{V}$) and represent a core tool for NLP [80]. Seq2Seq Neural Networks can be used to learn probabilistic language models: we can train a deep Neural Network to output the probability of a sequence of tokens as the product of the (conditioned) probability of the individual tokens in a sequence. Recent research showed that training *neural language models* (i.e., deep Neural Networks trained as language models) on large amounts of text data allows us to: (i) generates high-quality text (ii) yield very informative features (in the form of contextual embeddings) to be used for discriminative tasks (iii) later fine-tuning with state-of-the-art results on a downstream (generative or discriminative task) [22, 46, 137]. In general, these networks are trained to minimise the negative log-likelihood of the output sequence $P_{LM}(w_1, \ldots, w_i, \ldots, w_n; \vartheta)$.

Approaches. Neural Networks can be used to learn and approximate different language modelling approaches: *causal*, *bi-directional*, and *transducer* (see Figure 15). The approach to language modelling is a result of how the hidden transformation is computed. However, independently of this choice, the end-to-end behaviour of yielding a probability distribution is unchanged.

[5]Knowledge graphs are graph-based representations of complex structured and unstructured information.
[6]Speaker persona is the description of the unique characteristics, such as background and speaking style, that characterise an individual.

We talk of *causal* language models or *auto-regressive* language models or *decoder (only)* language models when the LM computes the probability of observing each token in a sequence $X = \langle x_1, \ldots, x_i, \ldots, x_{|X|} \rangle \in \mathcal{V}^{|X|}$ given only the preceding ones (see Figure 15(a)):

$$P_{causal\ LM}(X) = \prod_{i=1}^{|X|} P(x_i | x_1, \ldots, x_{i-1}) = \prod_{i=1}^{|X|} P(x_i | X_{i'<i}). \tag{25}$$

These models are trained on tasks like *causal language modelling* (predict next token given the preceding one) [22, 135].

We talk of *bi-directional* language models or *auto-encoder* language models or *encoder (only)* language models when the LM computes the probability of observing each token in a sequence $X = \langle x_1, \ldots, x_i, \ldots, x_{|X|} \rangle \in \mathcal{V}^{|X|}$ given all the tokens present in the sequence, the conditioned probability can be computed on a (possibly) corrupted copy of the original sequence $\widetilde{X}$ (see Figure 15(b)):

$$P_{bi\text{-}directional\ LM}(X) = \prod_{i=1}^{|X|} P(x_i | \tilde{x}_1, \ldots, \tilde{x}_{|X|}) = \prod_{i=1}^{|X|} P(x_i | \widetilde{X}). \tag{26}$$

These models are trained on *masked language modelling* (predict the missing tokens from a corrupted input sequence, similar to the denoising auto-encoders objective) [46, 103].

Finally, we talk of *transducer* language models or *encoder-decoder* language models when the LM outputs the posterior causal probability of a target sequence $Y = \langle y_1, \ldots, y_j, \ldots, y_{|Y|} \rangle \in \mathcal{V}^{|Y|}$ given a separate source sequence $X = \langle x_1, \ldots, x_i, \ldots, x_{|X|} \rangle \in \mathcal{V}^{|X|}$ (see Figure 15(c)):

$$P_{transducer\ LM}(Y|X) = \prod_{j=1}^{|Y|} P(y_i | X, y_1, \ldots, y_{j-1}) = \prod_{j=1}^{|Y|} P(y_i | X, Y_{j'<j}). \tag{27}$$

These models are trained on tasks like *prefix language modelling* (similar to causal language modelling, but the first elements of the sequence are visible to the model and they are not used to compute the loss), *span replacement* (predict the missing sub-sequences of token from the source), or *de-shuffling* (re-order the input sequence of tokens) [90, 137].

Note that Causal LM models have as output sequence the same input sequence shifted to the left and Bi-directional LM has as output sequence the input sequence with the same alignment (no shifting in any direction). However, the input to Bi-directional LM can be a corrupted version of the output. We underline this concept in Figure 15(b) using $\widetilde{X}$ ad input and X as output. Examples of causal LMs are *GPT* [22, 120, 134, 135], *Bloom* [151], *Gopher* [136], *Chincilla* [71], and *LaMDA* [183]. While, examples of bi-directional LMs are *ELMo* [127] *BERT* [46] or *RoBERTa* [103]. When implemented with Transformer networks, these two approaches to language modelling adopt, respectively, a fully visible masking pattern and causal masking pattern for their self-attention transformations.

On the contrary, Transducer LM work with two separate and orthogonal sequences (the source and the target sequences, respectively X and Y) that are both part of the input (the source is the input of the encoder and the target the input of the decoder), but only the target sequence shifted to the left is part of the output. *BART* [90], *T5* [137, 208, 209], *T0* [148], and *FLAN* [194] are all examples of transducers LMs. The shifting of the target is due to the autoregressive nature of the decoder in the transducer. In fact, when implemented with Transformer networks, this transducer language model, can be obtained either combining an encoder with fully visible attention and a decoder with causal attention using a fully visible cross-attention in the middle, or with a *non-causal decoder* [193].

In the context of ***Dialogue Language Modelling*** (**DLM**) (i.e., language modelling for dialogue) we consider a dialogue X under two perspectives: either as a plain sequence of tokens or a sequence

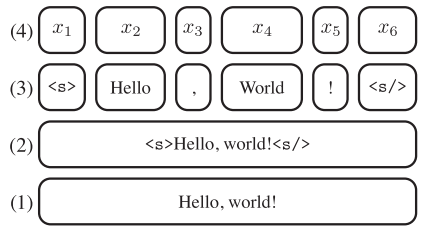


Fig. 16. Text encoding process. (1) Initial text string. (2) Pre-processed text string with additional tokens to mark begin and end of a sequence (optional, sometimes other special tokens like *separator* or *classification* are inserted [46]). (3) Tokenisation of the whole text sequence (either word or sub-word level tokenisation). (4) Conversion of each token to its index.

of n_X utterances, each representing a sequence of tokens on its own:

$$X = \langle x_1, \ldots, x_t, \ldots, x_{|X|} \rangle = \langle U_1, \ldots, U_i, \ldots, U_{n_X} \rangle, \tag{28}$$

where

$$U_i = \langle x_{i,1}, \ldots, x_{i,j}, \ldots, x_{i,|U_i|} \rangle, \tag{29}$$

with $x_t, x_{i,j} \in \mathcal{V}$. Note that given this notation, since the tokens in the plain sequence map bijectively with the tokens in the sequence of utterances, we have that $x_{1,1} = x_1$ and $x_{n_X,|U_{n_X}|} = x_{|X|}$

From this utterance level division, we can extract all the available context-response pairs:

$$X = \langle U_1, \ldots, U_i, \ldots, U_{n_X} \rangle \rightarrow \langle (C_1, R_1), \ldots, (C_i, R_i), \ldots, (C_{n_X}, R_{n_X}) \rangle, \tag{30}$$

where $U_i \in \mathcal{V}^{|U_i|}$ is a sequence of tokens representing a turn in the dialogue, $C_i = \langle U_1, \ldots, U_{i-1} \rangle = U_{i'<i}$ is the *context* associated to the ith turn in the dialogue and $R_i = U_i$ is the ith *response* (or turn) in the dialogue, with

$$R = \langle r_1, \ldots, r_i, \ldots, r_{|R|} \rangle, \tag{31}$$

$$C = \langle c_1, \ldots, c_j, \ldots, c_{|C|} \rangle, \tag{32}$$

where $r_i, c_j \in \mathcal{V}$. In Section 3 we detail how the aforementioned language modelling approaches are currently adapted for the dialogue task.

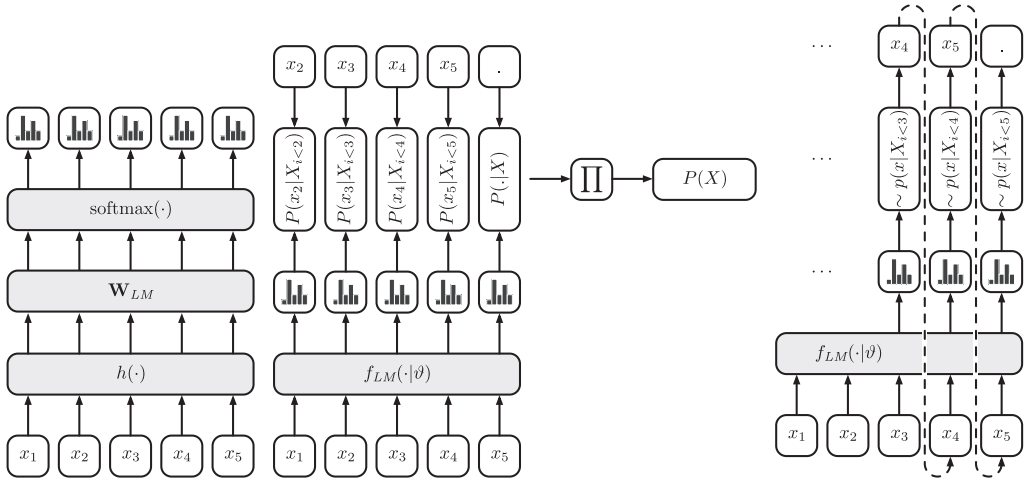
Text processing. All the Seq2Seq Neural Network models for language modelling share the same high-level architecture, as depicted in Figure 17(a): there is an input embedding layer to encode the sequences and transform them from sparse to dense representations, the hidden transformation layers compute the hidden representation of the sequences, and, finally, the output layer yields the posterior probability of observing a token sequence $\langle w_1, \ldots, w_i, \ldots, w_n \rangle$ (with $w_i \in \mathcal{V}$). The input sequence of tokens is extracted as in Figure 16.

For each output step, the Seq2Seq outputs a discrete probability distribution. Starting from this probability distribution, it is possible to apply decoding or sampling to generate text. All the inference uses of these models are visualised in Figure 17.

Independently of the modelling approach, any Seq2Seq model computes the output probability of a sequence as

$$f_{LM}(w_1, \ldots, w_i, \ldots, w_n; \vartheta) = P_{LM}(w_1, \ldots, w_i, \ldots, w_n) = \prod_{i=1}^{n} \text{softmax}\left(\mathbf{W}_{LM}^{\top} \cdot \mathbf{h}_i\right)_{w_i}, \tag{33}$$

where $\mathbf{h}_i \in \mathbb{R}^d$ is the contextual embedding corresponding to position i of the output ($1 \leq i \leq n$) computed through the hidden transformations $h(\cdot)$ of the Seq2Seq network (d is the size of the hidden representation), $\mathbf{W}_{LM} \in \mathbb{R}^{d \times |\mathcal{V}|}$ is the linear projection layer to compute the *logits* (i.e., the unnormalised log-likelihoods), and softmax$(\cdot)$ is the normalised exponential function. Notice that the softmax$(\cdot)$ outputs a vector of $|\mathcal{V}|$ elements, that is the discrete probability distribution



(a) Computing output distribution.

(b) Scoring sequence probability.

(c) Generating sequence sampling output distribution.

Fig. 17. High-level language model inference (example considering a causal LM).

over the possible tokens, we retain the w_i-th element to have the probability of the token in that position.

The input embedding layer takes care of projecting each token into a continuous vector space (the process is depicted in Figure 19). This representation is then transformed by the hidden layers. In more recent Transformer models, the input includes position embeddings, to take into account positional information [186].

The output layer is a final linear transformation followed by a softmax(·) activation. This final transformation is highly demanding in terms of computation costs, due to the high dimensional size of the output. In fact, the projection matrix is $\mathbf{W}_{LM} \in \mathbb{R}^{|\mathcal{V}| \times d}$, where d is the dimension of the hidden feature vectors and $\mathcal{V}$ is potentially large. In fact, before the introduction of sub-word tokenisers [87, 156, 204] which reduced considerably the value of $|\mathcal{V}|$, it was common practice to constrain $\mathcal{V}$ to the most frequent tokens [92], or substitute the softmax(·) activation with its hierarchical variant [113].

The input layer and the final output layer are linear projections whose dimensions have the same semantic meaning. Taking advantage of this aspect, many models rely on *weight tying* (or *weight sharing*) [78, 132], using the same parameters for the embedding and output layers. In this way, the number of parameters is considerably reduced.

The hidden transformations are the actual Seq2Seq Neural Network. The choice of the hidden transformation directly influences the language modelling approaches. Unidirectional (forward) recurrent networks and self-attention transformers with causal attention mask pattern is used to build causal language models [22, 135]. Bi-directional recurrent networks and self-attention transformers with fully visible attention-mask patterns are used to build bi-directional language models [46, 103]. Encoder-decoder recurrent networks, encoder decoder-decoder transformer networks or non-causal transformer networks with prefix mask pattern are used to build transducer language models [90, 137].

3 BASIC ARCHITECTURES

In this section, we present suitable Seq2Seq architectures that can be used to implement generative chatbots. These models build up on probabilistic language models introduced in Section 2.2.2,

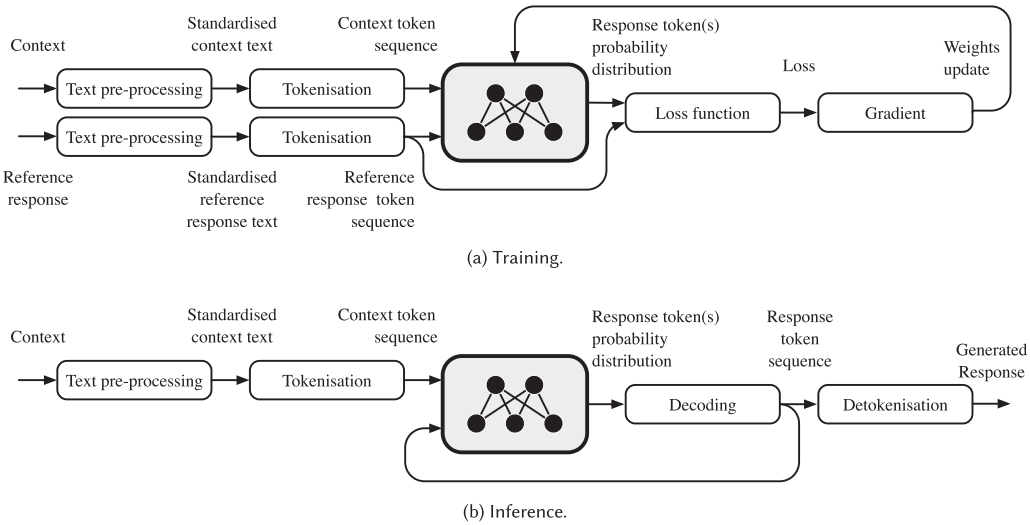


Fig. 18. Seq2Seq model high-level pipeline visualisation.

for this reason, we often talk about *Dialogue Language Models/Modelling* (DLM). In general, deep generative chatbots model the probability of observing a response $R = \langle r_1, \ldots, r_i, \ldots, r_{|R|} \rangle$ (the current utterance in a dialogue) given the context $C = \langle c_1, \ldots, c_j, \ldots, c_{|C|} \rangle$ (the concatenation of the utterances in a dialogue prior to the response) –with $r_i, c_j \in \mathcal{V}$, the vocabulary– as a conditional probability $P_{DLM}(R|C)$.

Independently of the actual architecture, there are two basic input/output pipeline flows, as for any other Neural Network: *training* and *inference*.

As reported in Figure 18, when used at training time, the model takes the context C (i.e., in general, previous information about the state of the conversation like the previous turns) and the target response R as input, and generates a distribution over the response tokens; such output distribution is matched against the target response to compute the loss and update the model's weights.

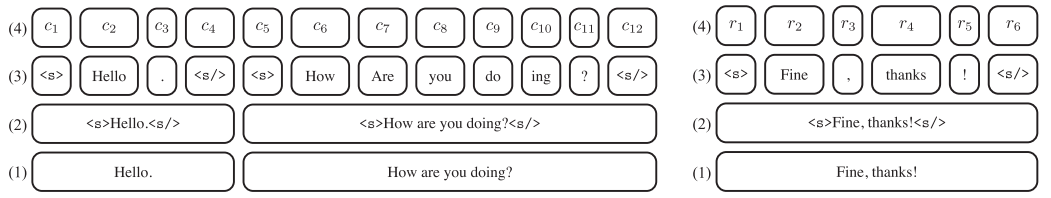
When used at inference time, the model takes only the context C as input and yields the output distribution over response tokens, from which the predicted response $\widehat{R}$ is derived by means of decoding (and then possibly used as an additional input), as we explain in Section 4.2.

DLMs follow the same preparation and processing steps that apply to generic language models. We visualise the process again in Figure 19 to account for the differences due to the dialogue structure.

3.1 Base Models

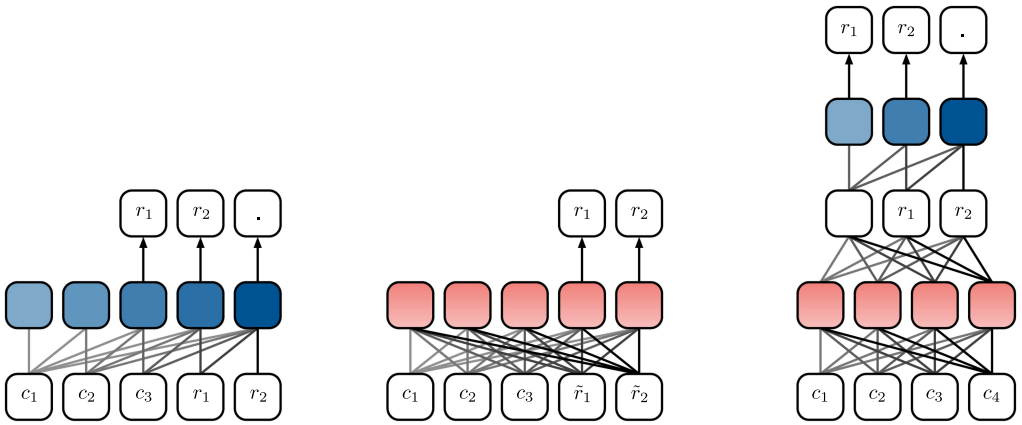
The most important part of a model is given by its set of hidden transformations, which characterise the entire architecture and affect how the output is computed (see Figure 20). In the following, we describe the three main approaches to designing the architecture of the agent (and thus its hidden transformations). Notice that these concepts do not depend on the configuration of the Neural Network: training or inference.

3.1.1 Causal Decoder. Causal models are the oldest architectures (see Figure 20(a)). These chatbots process context tokens and response tokens as a single sequence one after the other, shifting information from left to right, at each step. When processing the response, the input of a given step includes the output token generated at the previous one. Thus, due to such an



(a) Context (first turn is the agent's previous response, then the user follows). (b) Response.

Fig. 19. Dialogue encoding process for the context and the agent's new response. (1) Initial text string. (2) Pre-processed text string with additional tokens to mark begin and end of turns (optional). (3) Tokenisation of the whole text sequence. (4) Sub-word tokenisation of the resulting tokens from previous step (optional). (5) Conversion of each token to its index. Optional steps depend on the implementation.



(a) Causal decoder. (b) Bi-directional. (c) Transducer.

Fig. 20. Seq2Seq model architectures variants for dialogue. Notice that these figures are simplified representations of the architectures to explain data processing and do not correspond specifically to train or inference configurations.

autoregressive structure, the posterior probability of any token in the response is computed considering the context and all the preceding tokens (see Equation (34)).

$$P_{DLM}(R|C) = \prod_{i=1}^{|R|} P(r_i|C, R_{i'<i}). \tag{34}$$

This architecture was initially implemented using uni-directional RNNs [92, 144, 189]. Nowadays, Transformer blocks are used for the same purpose [99, 202, 216], where the self-attention transformation employs a causal attention mask pattern to prevent the attention from considering "future" tokens.

This causal decoder dialogue language model is the most common approach to Seq2Seq chatbots. As examples, see *ChatGPT* [118], *Bard* [63], *Bing Chat* [111], *Gopher* [136], the current version of *BlenderBot* [86], *TransferTransfo* [202], *CAiRE* [99], and *DialoGPT* [216], which are all Transformer network-based chatbots.

3.1.2 Bi-directional Encoder. In general bi-directional encoder models are used to build discriminative models for classification and regression, rather than generative ones. However, this kind of architecture allows for generative solutions too. In fact, auto-encoders have been employed for

neural machine translation [65, 81, 89, 96], and generic text generation [192]. Thus, auto-encoders can be considered suitable architectures for generative chatbots [76].

Similarly to causal solutions, auto-encoder chatbots process in sequence context tokens and response tokens as part of the same sequence. Unlike causal chatbots, however, all the tokens are processed in parallel rather than in sequence.

The most straightforward solution is to start from some initial blank representation[7] of the response $\widetilde{R}$, and then fill in parallel these blank response positions, as depicted in Figure 20(b) (note the two empty boxes). The resulting model starts from such blank tokens and computes for each of them, in parallel, the response token distributions. This transformation can be repeated multiple times, refining the generated distribution. This behaviour is similar to that of a *denoising auto-encoder* [89]. There are other approaches, but they require complex decoding schemes [96, 192] or the use of hierarchical models [89, 192] (we present the hierarchical group in Section 3.2).

In general, in non-autoregressive models like auto-encoders, the posterior probability of observing one of the response tokens is considered independent of the other response tokens. However, as shown in Equation (35), the probability of observing the ith response token depends on all the context and response tokens. In fact, the hidden representation in correspondence of the tokens in $\widetilde{R}$ is built using all the input positions, and the output probabilities are computed from those hidden vectors.

$$P_{DLM}(R|C) = \prod_{i=1}^{|R|} (r_i|C, \widetilde{R}). \tag{35}$$

This chatbot architecture can be implemented using bi-directional RNNs or Transformers with fully-visible attention masking patterns. However, this approach is rarely used since encoder models are usually employed for discriminative tasks, rather than generative ones. Thus, these models perform better on tasks like *retrieval question answering* [69].

Note that, since for each response the number of blank tokens to provide as input must be known in advance, the length of such response must be set a priori. Alternatively, it is possible to extend the Neural Networks with a module able to predict the output length [89]. This length can be used for adding the required blank tokens.

3.1.3 Transducer. The transducer models combine ideas from the two previous architectures and were initially introduced for machine translation [7, 179, 186].

These chatbots split the context encoding step (the red part in Figure 20(c)) from the response generation one (the blue part in Figure 20(c)). An alignment module between these two parts is needed (the connection between encoded C and R of Figure 20(c)), to leverage the encoded context information during the response generation step. In Transformer network-based implementations, use a fully visible masking pattern, while the decoder self-attention uses a causal masking pattern.

Transducer, sometimes *encoder-decoder*, chatbots are designed to process the response in an autoregressive way. Thus, the posterior output probability of observing the response is the same as the one of the causal models (see Equation (34)). Unlike causal and auto-encoder models, in encoder-decoder approaches the hidden representations of context and response do not necessarily share the same vector space. In fact, context and response are encoded separately, and then some alignment strategy (e.g., cross-attention) is used to align them.

This architecture has been implemented with RNNs, usually relying on a unidirectional approach, for both encoder and decoder [93]. However, the encoder could be realised with a bi-directional RNN [166]. Then, the hidden states generated by the encoder are just passed to

[7]This blank token can correspond to an embedding of all zeros, a learnt mask of some random vector.

the decoder, realising the alignment. An example of chatbots using this approach with RNNs is *XiaoIce* [220].

More modern approaches are based on Transformers with a fully-visible attention masking pattern in the encoder self-attention and decoder cross-attention and a causal masking pattern in the decoder self-attention (the latter is to achieve autoregressive generation), like *Meena* [1] or the first version of *BlenderBot* [145]. The additional cross-attention block is used to align the current response token (i.e., the attention query) with the input context (i.e., the keys and values of the attention). Alternatively, there are some examples of architectures adopting prefix mask patterns (the non-causal architectures), where full attention is used on the context, and causal attention on the response [9, 10].

3.2 Hierarchical Models

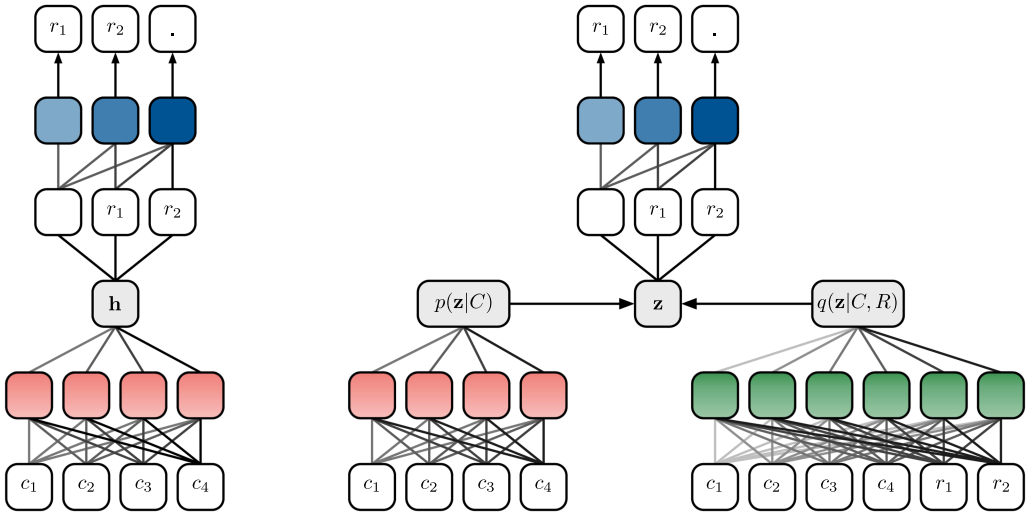
Human language has an inherent *hierarchical structure* [59, 159, 188]. Some models have been proposed to capture this hierarchical aspect in conversation, like Ventola's [188] or the DAMSL [4]. One of the high-level aspects of human language, during conversations, is the so-called *dialogue act*, which represents the speaker's intention and the effect it has on a listener. For example, a question, a statement, or a request for action. This hierarchical structure enables humans to reason on high-level aspects, like the desired dialogue act, before producing the utterances that realise it.

This hierarchy has been approximated in neural conversational agents (both open-domain and task-oriented) to improve them. We distinguish between simple *latent hierarchical* models and *variational latent hierarchical* models. These models can be trained using either continuous [159, 160] or discrete [10, 149] latent vector representations.

3.2.1 Latent Hierarchical Models. Latent hierarchical Seq2Seq models have been introduced to improve the chatbot's dialogue modelling capabilities with the goal of helping them to manage longer conversations. These architectures represented a significant improvement for models based on recurrent networks, as they couldn't manage very long contexts. In particular, latent hierarchical Seq2Seq models cope with longer conversations by introducing explicit components to manage the high-level aspects of the conversation (the green part in Figure 21(a)). Although, nowadays, attention-based models handle long conversations in a better way (removing the need of explicit components to manage the context at a high-level) latent hierarchical Seq2Seq models are still used for conditioning the generation on high-level aspects. Note, however, that such aspects are usually explicit –in the sense of human understandable– rather than latent, and thus latent hierarchical Seq2Seq models are rarely applied (more on this in Section 4.3).

The core idea of hierarchical chatbots is to predict a hidden latent representation, encoding the entire response into a single vector (a *turn embedding*), and then proceed with the token sequence generation exploiting such hidden latent representation. Referring to Figure 21(a), the latent representation is the grey box, derived from the context which is used to guide the response generation, as if it was a "compressed" response that the decoder "expands". During training, the chatbot learns the implicit high-level dialogue model, which predicts the latent response embedding (i.e., a latent representation of the entire chatbot's response), and the explicit low-level model, which predicts the responses' tokens.

Early solutions of this kind adopted RNNs and an encoder-decoder approach [175], like the *HRED* models [159] and *XiaoIce* [220], the latter uses the hierarchical approach to condition the response generation on its empathetic-response targets. Given a turn of the conversation, the encoder part of the Seq2Seq model extracts a single hidden vector representing the entire turn and passes it to a high-level recurrent network. This high-level network is realised through a separate RNN working directly on latent representations: the input is the sequence of turn embeddings and



(a) Latent model. (b) Variational latent model.

Fig. 21. Seq2Seq hierarchical model architectures. Notice that these figures are simplified representations of the architectures to explain data processing and do not correspond specifically to train or inference configurations.

the output is the response turn embedding at each time step (again, the grey box of Figure 21(a)). The predicted turn embedding is used to condition the low-level generation in the decoder by concatenating it to each response token embeddings.

In theory, the hierarchical approach also applies to Transformer networks (it has been adopted for task-oriented dialogue [66, 150]). Moreover, it also works on architectures other than the encoder-decoder. Finally, it is worth to mention that it is also compatible with discrete representations, not only continuous ones [149].

3.2.2 Variational Hierarchical Models. The hierarchical models were quickly extended into *variational* solutions (see Figure 21(b)), similarly to the ***Variational Auto-Encoders*** **(VAEs)** [84, 185]. In VAEs the hidden representation is a random variable $\mathbf{z}$, this variable is sampled using the *reparametrisation trick* to make the entire process differentiable.

At training time, the hidden latent representation $\mathbf{z}$ is sampled from the *approximated posterior distribution* $q(\mathbf{z}|C,R)$ ($q(\mathbf{z}|C,R) \approx p(\mathbf{z}|C,R)$). At inference time (i.e., when predicting a new response), instead, the *prior distribution* of the latent given the context $p(\mathbf{z}|C)$ is used in place of the posterior to sample $\mathbf{z}$. In both cases, the latent $\mathbf{z}$, possibly with the context, is used to condition the generation of the response from its likelihood $p(R|C,\mathbf{z})$.

Despite being conceptually similar to vanilla hierarchical chatbots, variational solutions have a regularised hidden space for the latent representation $\mathbf{z}$, which enables an easy *exploration of the latent space* allowing for the generation of *diverse responses* [9, 10, 218]. Moreover, while vanilla versions do not alter the training process, the variational models require a slightly different objective to optimise (more on this in Appendix A.1).

NLP has already explored this technique for diverse text generation, using variational latent space representations [21, 217]. Early chatbots implementing these approaches were still based on recurrent models, like *VHRED* model family [158, 160, 218], since they were the direct extensions to the previous hierarchical chatbot models. As before, this variational approach is compatible with Transformer networks; some of such networks even adopted a discrete latent representation

(see *PLATO* [9, 10]) to reduce the variance of latent space and produce more interpretable representations.

4 ADVANCED TOPICS

Up to now, we have described architectures that leverage probabilistic language models as generative chatbots. There are further aspects to consider when using these architectures. In particular, *hybrid models*, *decoding*, and *conditioning*.

In the probabilistic language models, we presented so far as generative dialogue models, decoding is essentially sampling the output probability distribution yielded by these models and generating a response. But, in some cases, we want the generated text to have certain properties or attributes. In this sense, we introduce the concepts of *conditioning* (or *controlled decoding*) and *prompting*. Moreover, the models introduced up to now only rely on generative mechanisms, while, in some cases, having a hybrid model capable also of retrieval approaches may improve the general quality.

4.1 Hybrid Models

Some solutions for generative chatbots showed improvements when combining the retrieval approach with the generative one. In particular, we distinguish between *multi-objective* models and *retrieve-and-refine* models.

4.1.1 Multi-objective. Multi-objective models use a single architecture that combines the two approaches to implement data-driven chatbots: generative and a retrieval [9, 10, 202]. The idea is to leverage two distinct heads on top of the hidden representations, one to do the usual language modelling and the other to classify the responses into *correct* $C_{correct}$ and *incorrect* $C_{correct}$ ones.

This additional retrieval head implicitly learns a scoring function by predicting the matching probability of the context and a response (i.e., the probability of a given response to belong to the correct responses class $C_{correct}$). This scoring function can be used to rank possible responses: responses more suitable for the given context should have an high probability score and responses not suitable for the given context should have a low probability score. We call them *retrieval* chatbots because this scoring function can be used to search a data set of possible responses for the best match (the highest-ranked response with respect to the current context in the collection).

The retrieval head $f_{CLS}(\cdot)$ computes the posterior probability of a response to belong to $C_{correct}$, as in Equation (36):

$$P_{CLS}(R \in C_{correct}|C) = f_{CLS}(C, R) = \sigma\left(\mathbf{w}_{CLS}^{\top} \cdot \mathbf{h}_{CLS}\right), \tag{36}$$

where $\sigma(\cdot)$ is the *sigmoid function*, $\mathbf{h}_{CLS} \in \mathbb{R}^d$ is the hidden vector representing the sequence to classify and $\mathbf{w}_{CLS} \in \mathbb{R}^d$ is the vector of learnable weights of the retrieval head.

Since we have a binary classifier, we can compute the probability for a response to be correct, and then derive the probability to be incorrect from the first one, as in Equation (37). Thus, the idea is to obtain a high probability for correct responses R_+ and a low probability for incorrect responses R_-.

$$P_{CLS}(R \in C_{incorrect}|C) = P_{CLS}(R \notin C_{correct}|C) = 1 - P_{CLS}(R \in C_{correct}|C). \tag{37}$$

Given a conversation corpus, an incorrect response (sometimes called *distractor* or *contrastive* sample) can be easily obtained sampling a random substitute response from the corpus. Moreover, some solutions have been proposed to avoid raw sampling and leverage semantic similarity between contexts or responses when selecting such distractors [27].

During training, as we explain in Appendix A.1, the model uses a combination of language modelling and retrieval objectives. During inference, the two heads can be either used independently or it is possible to re-rank the responses generated by the language modelling head, according to the probability predicted by the retrieval head [9, 10].

Examples of models using this multi-objective approach are *TransferTransfo* [202], *CAiRE* [99], and *PLATO* [9, 10]. CAiRE also uses an additional objective to predict user's emotion.

4.1.2 Retrieve-and-refine. Retrieve-and-refine models were introduced to cope with two issues of basic generative models: the generation of dull responses and the so-called "hallucination of knowledge" [86, 145, 198, 206] (the latter refer to the case where agents generate responses without actually knowing the information it is talking about or without referring to some knowledge base, leading to possibly wrong/misleading information). This approach aims at generating the response starting from the context sequence and an additional sequence, selected by a retrieval model, that can be either a possible response [198], or some external knowledge retrieved from the web [86].

Given the context of the conversation, it is possible to leverage a pre-trained retrieval model to select a candidate response $R_{candidate}$ from a corpus, or a segment of a document K containing the knowledge necessary to generate the answer. These sequences can be appended to the context and provide additional information during the generative step. The output response probability becomes thus $P(R|C, R_{candidate})$ or $P(R|C, K)$ depending on the approach: retrieve and refine or knowledge-grounded.

The generative model can exploit the candidate response $R_{candidate}$ to yield an utterance showing a more vibrant language, typical of human-generated responses [145]. Similarly, the generative model can exploit the additional knowledge K to ground the response on actual information instead of "hallucinating" it using the knowledge embedded in its weights [145]. Notice that the two approaches are not alternative: $P(R|C, K, R_{candidate})$.

Despite retrieve-and-refine models showed to improve over vanilla generative models [198], training these models is difficult since they do not always learn to exploit the additional information in a proper way [145]. However, solutions have been proposed to cope with this issue by randomly alternating the retrieved response and the target one [145]. The first version of *BlenderBot* [145] is an example of correctly trained retrieve-and-refine model.

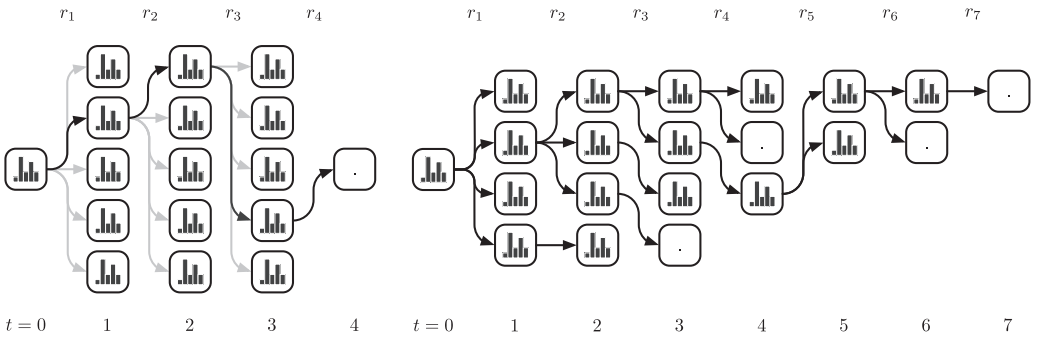
An important point to keep in mind is that at training time it is necessary to have the retrieved sequence. While this requirement does not represent an issue for response retrieval models –since the retrieval model can be trained on a generic conversation corpus (even the same one)– this is an issue for knowledge retrieval systems. In fact, in this case, it is necessary to have a *gold knowledge*, retrieved by humans, to be sure that the selected segment of text is relevant to the context and contains information necessary to generate the response.

4.2 Decoding

All the presented models do not directly output a sequence of tokens, but rather a sequence of probability distributions over the vocabulary $\mathcal{V}$. These sequences need to be decoded somehow to extract the actual token sequence composing the response. There are multiple ways to accomplish this task.[8]

Since the Seq2Seq are trained to maximise the probability of the correct response, the best-generated response should be the most probable one (which maximises the expression in Equations (34) and (35)). Ideally, we would like to use an *exaustive search* algorithm to decode

[8]In the following we refer to an autoregressive decoder, but the same concepts apply to other models.



(a) Greedy. (b) Beam search decoding with beam size 4.

Fig. 22. Deterministic decoding.

the output probabilities and retrieve the most probable sequence. In practice, this is unfeasible due to the size of the output space (the branching factor is the size of the vocabulary $\mathcal{V}$). Thus, approximations exists that allow exploration of the output space [72].

In general, we distinguish between *deterministic* and *stochastic* decoding. In deterministic decoding the next token is sampled according to some fixed rule and always yields the same output. In stochastic decoding the next token is sampled randomly from the distribution predicted by the language model. These techniques can be combined with approaches to explore multiple possible response sequences.

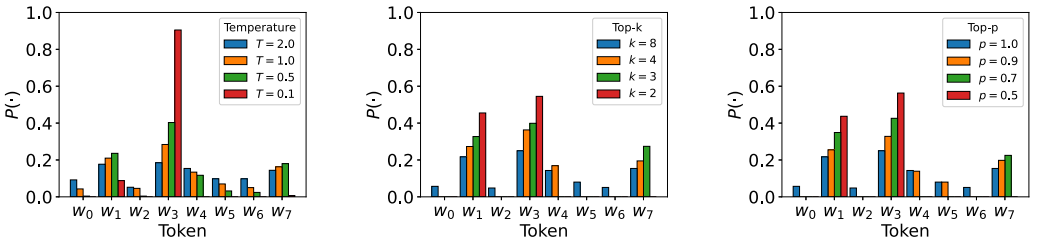
4.2.1 Single Sequence. The most straightforward decoding approach is called *greedy decoding* where, at each step, the most probable token is fed back to the model, as in Figure 22(a). However, simple greedy decoding usually leads to dull responses (e.g., "I don't know") that often contain degenerated text full of repetitions.

Alternatively, it is possible to resort to *multinomial sampling*. The idea is to sample a token according to the output probability distribution over the vocabulary $\mathcal{V}$, recur the token as the next response token, and then repeat the process iteratively until some stop criteria (e.g., the end-of-sequence token is sampled). This can help avoid the bland, and incoherent text sometimes yielded by maximisation-based decoding methods like greedy decoding [72]. The process can be repeated to sample multiple responses and retrieve the most probable one.

Some variations to vanilla sampling can be combined. The *temperature re-scoring* is applied on the logits (hence, before the exponential normalisation of the softmax$(\cdot)$), dividing the logit scores by a temperature parameter τ, as in Equation (38). If $\tau < 1$ the distribution gets more "peaky" and if $\tau > 1$ the distribution gets more "smooth". Usually, a $\tau < 1$ is used to reduce the probability of more unlikely tokens being sampled [1]. Examples of application of temperature re-scoring to the output distribution can be found in Figure 23(a).

$$P(x_t) = \text{softmax}\left(\left(\mathbf{W}_{LM}^{\top} \cdot \mathbf{h}_t\right)/\tau\right)_{x_t} = \frac{\exp\left(\left(\mathbf{w}_{x_t}^{\top} \cdot \mathbf{h}_t\right)/\tau\right)}{\sum_{i=1}^{|\mathcal{V}|} \exp\left(\left(\mathbf{w}_i^{\top} \cdot \mathbf{h}_i\right)/\tau\right)}. \tag{38}$$

Another common sampling approach is filtering on the best candidates before sampling. In this sense, two complementary approaches exist: *top-k* and *top-p* sampling (the latter is also referred to as *nucleus sampling*). Top-k prescribes to consider only the most probable $k \in \mathbb{N}^+$ tokens (where k is predefined) and sample among them according to their probability. Instead, top-p prescribes to consider the smallest set of tokens such that the sum of their probability is $\geq p \in [0, 1] \subset \mathbb{R}$ (thus, yielding a variable number of possible tokens) [72]. Examples of application



(a) Temperature. (b) Top-k. (c) Top-p.

Fig. 23. Effects of sampling strategies on the token probability mass.

of top-k and top-p re-scoring to the output distribution can be found, respectively, in Figure 23(b) and 23(c).

Finally, a very recent solution is *contrastive search* [178]. This deterministic approach combines top-k with a re-scoring function: the next token is selected among the first top-k that maximises a weighted score combining the token score and a degeneration penalty. The score is computed as in Equation (39), where $\alpha \in [0, 1] \subseteq \mathbb{R}_0^+$ is a tunable hyperparameter, and $s(\cdot)$ is the cosine similarity.

$$x_t = \underset{\hat{x} \in \text{top-}k(p(x|x_{t'<t}))}{\arg\max} \{(1-\alpha) \cdot P(\hat{x}|X_{t'<t}) - \alpha \cdot \max_{t' \in [1,t) \subseteq \mathbb{N}} s(\mathbf{h}_{\hat{x}}, \mathbf{h}_{t'})\}. \tag{39}$$

With anisotropic models[9] this decoding scheme allows to generate very diverse responses. In fact, at each step the decoding scheme selects the token with the most different latent representation among the k most probable tokens.

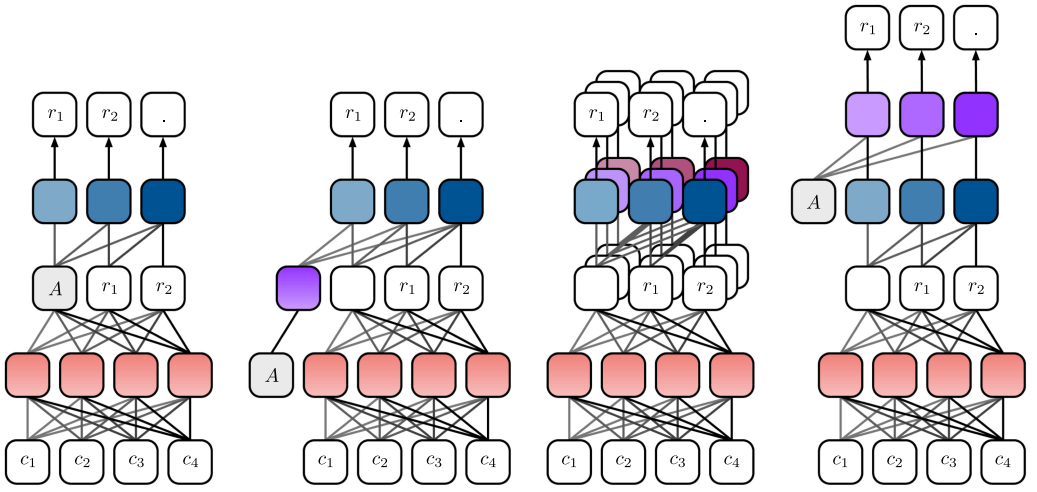
4.2.2 Multiple Sequences. When decoding the response sequence it can be useful to generate multiple candidates and then select among them. An exhaustive search is not feasible, yet there are some available approximations.

The deterministic or stochastic decoding schemes for single sequences can be combined with an orthogonal technique: *beam search decoding*. Beam search decoding is a heuristic that considers multiple candidates while decoding, yielding multiple sequences in output. The basic idea is to keep track of a fixed number of candidates (beam) and carry on their decoding, in parallel. The decoding process is advanced for each candidate at each step, keeping the best new elements. All these new candidate sequences are then re-ranked on the new cumulative probability. Only up to the first n sequences (with n being the so-called *beam size*) are kept, as depicted in Figure 22(b).

A simpler alternative is to *sample and re-rank*. The probabilistic language model is used to sample independently multiple sequences. These sequences can be re-ordered according to some metric or scoring function. For this purpose, leveraging a multi-objective model, which can use the retrieval head to score the candidates, can be helpful.

In this re-ranking settings, it is worth to mention the *Maximum Mutual Information* scoring approach [91, 216]. The idea behind this technique is to employ a separate *backward model* to re-score generated candidates. The backward model tries to predict the probability of observing the context (or, at least, the latest utterance) given the response. In this way, keeping the response that maximises the context probability would help filter out bland and inconsistent responses. Notice that this technique, despite being useful, requires the training of a completely separate model that must be used to re-rank all the generated responses, increasing the overall computational cost.

[9]In this context, *anisotropy* is a measure of how well embeddings are "spread" in their hyperspace. Usually, language models with more than 500M parameters are naturally anisotropic [178].



(a) Prepending. (b) Concatenating. (c) Mixture. (d) Adapter.

Fig. 24. Conditioning strategies. Notice that these figures are simplified representations of the architectures to explain data processing and do not correspond specifically to train or inference configurations.

4.3 Conditioning and Prompting

Conditioned (or *controlled*) generation is a relevant feature to include in a chatbot. In fact, these agents can benefit from following some high-level behaviours. Thus, conditioned generation is fundamental to controlling the generated response's content and style. This control, ideally, enables selecting different aspects of the generated text, from the simpler ones, like the emotion or the dialogue act, to more complex ones, like the agent's persona or knowledge ground. Apart from the possible architectural changes and extensions to the SEQ2SEQ agent, to work correctly at inference time, conditioning may require one or more additional modules to predict the expected content of the response.

In general, neural models for conditioned sequence generation use a set of desired attributes $A \equiv \{a_1, \ldots, a_k\}$ to output the posterior probability of a sequence given these attributes. Maximising the output posterior probability will ideally lead to a sequence showing the properties encoded through the attributes. In the specific case of chatbots, the posterior probability is conditioned on the context, too (see Equation (40)). This target posterior probability distribution to approximate is similar to that of hierarchical models.

$$P(R|C, A) = \prod_{i=1}^{n} P(r_i|C, A, R_{i'<i}). \tag{40}$$

There are multiple approaches to train a conditioned generative chatbot. The most straightforward solution is to train the conditioning and the generative aspects, together, in an end-to-end manner. There are two major approaches of this kind: either *prepend the conditioning attributes* to the sequence to generate or *concatenate the conditioning embeddings* to the hidden text representation. The former approach, depicted in Figure 24(a), instructs the generative model through the input sequence. It is possible to insert "special" tokens representing the attributes (e.g., emotion or dialogue act), which will be converted into embeddings by the input layer [99, 117, 211]. Alternatively, it is possible to use entire text sequences describing the agent's persona or the knowledge necessary to generate the answer (these text sequences are usually marked through additional type tokens or "special" start and end tokens) [50, 99, 202], *TransferTransfo* [202] and *CAiRE* [99] use prepending to add persona-grounding, while *BlenderBot* [86] uses it for knowledge grounding.

The latter approach directly alters the hidden state of the model, as depicted in Figure 24(b). The embedding of the desired attribute (or attribute combination) is concatenated (or summed) to each of the initial word (or token) embeddings coming from the input layer, the resulting "extended" embedding is then used to feed the hidden transformations [92, 220]. For example, *XiaoIce* [220] uses the concatenation approach to condition the generation towards the desired attributes prescribed by its cognitive and emotional models.

Some solutions suggest using a *mixture* of generative models, where each one of them is specialised on a specific attribute (or attribute combination) [98] (see Figure 24(c)). For example, when conditioning on emotions, we will have a decoder for "happiness" (i.e., trained only with samples labelled as "happiness"), a decoder for "anger" (i.e., trained with samples labelled as "anger"), and so on. Depending on the selected emotion, the corresponding decoder is used to generate the conditioned response. Consequently, during training, only the weights of the decoder corresponding to the target attribute are updated on the given response. This mixture approach does not scale up well, since the generative portion of the network must be replicated for each possible attribute combination, and is not compatible with non-categorical conditioning attributes (e.g., persona or knowledge). The chatbot *MoEL* [98] uses this approach.

An alternative approach to end-to-end conditioning is to just learn the adaptive (conditioning) model. The key idea is to decouple the generative part from the conditioning one [106, 212]. In this way, the generative core chatbot has to be trained only once (which is the most onerous operation), and the conditioning modules can be trained and freely plugged into the core dialogue model, as in the architecture depicted in Figure 24(d). These adapter layers [11, 74] (the purple blocks in Figure 24(d)) are responsible of altering the hidden representations of the network to achieve the conditioning. Notice that, given the pre-trained generic chatbot, this technique requires training a smaller number of parameters than standard fine-tuning. In fact, this approach requires training only the conditioning layers, instead of fine-tuning the entire model. The *Plug-and-Play conversational model* [106] uses this conditioning technique. The advantage is that, by design, the adapter layers fewer parameters than the rest of the network. However, it is hard to achieve results as good as those of fine-tuning [137].

Other solutions involve using an *attribute discriminator* to carry out weighted decoding. It is possible to either re-score the output sequences using the attribute discriminator [155] or apply more sophisticated techniques of this kind as PPLM [41]. The advantage of these techniques is that they allow employing pre-trained chatbots and discriminative models to produce conditioned generative dialogue agents without needing specific training. The disadvantage is that the more complex decoding scheme slows down the inference. PPLM powered chatbots [2, 106] are more sensitive to this issue since they need to alter, step-by-step, the hidden state of the model in forward-backwards manner, making this approach hardly usable in real-time scenarios.

All these models rely on conditioning approaches and need, to be trained, conversational corpora labelled with the desired attributes. This requirement usually gets problematic due to possible unbalances among the attributes, or to unsatisfactory corpora size. To overcome this limitation it is possible to add synthetic labels, using a discriminator [171].

In the last three years, the development of *large language models* like *GPT-3* has introduced the concepts of *prompting* and *few-shots learning*, sometimes referred to as *in-context learning* [22, 120, 136, 148, 194], to condition the output of the model. The idea behind prompting is to use natural language to describe the desired behaviour or output as initial part of the input context, and then leverage model completion capabilities to generate an output that responds to the requests. For example[10]:

[10]The example is adapted from Gopher.

```
The following is a conversation between a highly knowledgeable and
intelligent AI chatbot and a human user. In the following interactions,
the user and the AI will converse in natural language. The AI chatbot
is built to be respectful, polite, inclusive and to do its best to
answer the user's questions.

User: Hi, I'm going to start by quizzing you with some questions.
      Who is captain Malcolm Reynolds?
AI:
```

For a sufficiently complex Seq2Seq language model trained on a large amount of data this is sufficient to start generating the response utterance.

In a sense, this prompting approach is a generalisation of the "prepending" approach to conditioning, where the attributes are described in natural language and may affect the entire behaviour of the agent, rather than a single response. Moreover, introducing examples of the desired behaviour on similar inputs as part of the context helps the model extrapolate and carry out the desired task. This is why we talk about few-shots: shots are the examples in the context, paired with the prompt to obtain the desired behaviour [22].

In the context of chatbots, prompting and few-shot learning can be used to instruct the model to follow a specific behaviour. Examples of models using these appraoches are *ChatGPT* [118], *Bard* [63], *Bing Chat* [111]. models like ChatGPT or Bing Chat rely on an additional ad-hoc training for chatting. In the case of the language model *Gopher*, instead, generic dialogue pre-training and prompting were sufficient to produce a language model capable of having conversations at the same level its ad-hoc trained version for dialogue [136]. Dropping the additional task-specific training may be an important key towards the future developments, which we discuss in Section 5.2.

5 CONCLUSION

In this section, we summarise the content of this article and we report our considerations on the possible future directions and open issues connected to generative Seq2Seq chatbots.

5.1 Concluding Remarks

In this article, we explained how to employ neural Seq2Seq architectures to build and train open-domain conversational agents (or chatbots). This kind of architecture allows the implementation of generative data-driven chatbots, which show more robust adaptive capabilities than retrieval based ones.

These Seq2Seq models work on text represented as a sequence of embeddings, and leverage recurrent layers, or Transformer layers with attention, to process the input (the context) sequence and yield the output (the response) sequence. Each time step output actually contains a probability distribution over the vocabulary words. Such time step distributions need to be decoded to obtain the final tokens composing the response.

The Seq2Seq models are mainly trained to maximise the probability of the target response in the sample conversation, given the context (i.e., the preceding turns). Many corpora exist for open-domain conversation, including large corpora for pre-training and smaller corpora for fine-tuning. Once trained, these models can be evaluated either through automatic metrics or by means of human raters. Moreover, these Seq2Seq can be further optimised some human feedback for a continuous improvement. Additionally, it is possible to join one of the available competitions to test the quality of the conversational agents, comparing it with others.

5.2 Future Directions

The landscape of NLP, in general, and the landscape chatbot develpement, in particular, are evolving quickly. *ChatGPT* [118], *Bard* [63], *Bing Chat* [111], and so on. have shown how impressive a large language model based on Seq2Seq architecture and trained on a massive amount of dialogues can be.

In the near future, we expect to see a further scaling of these models, to be more complex and to train on more data. Moreover, we expect these Seq2Seq models to become end-to-end multimodal models, thus allowing multiple forms of inputs and outputs, like images or audio, to be mixed with regular text. Multimodal agents are already being developed, like XiaoIce [220], Gato [141], or GPT-4 [120], but these functionalities are not always deployed with the chatbot (for example ChatGPT with GPT-4 backend, to this day, does not support image inputs yet despite GPT-4 does).

As models scale up, we expect also a convergence with task-oriented systems. Initial works in this direction adopted Seq2Seq architectures to train generative models for task-oriented conversational agents [23, 88, 102]. The idea was to fine-tune pre-trained language models on task-oriented dialogues data [24] adding, for example, specific tokens to represent the additional pieces of information for the dialogue agent or prepending text containing the description of the agent's belief state (i.e., the *dialogue state* according to the agent, usually expressed as a set of *slot-value pairs*). Tracking of the belief state is delegated to additional pieces of the Neural Network. These models generate a response starting from the description of the belief state and the context of the conversation. Currently, the general purpose capabilities of large language models used as chatbots are slowly allowing to use the same model for both open-domain and task-oriented dialogues, guiding the behaviour with the initial prompt description. Moreover, some services are starting to develop compatible plugins to exploit these chatbots, as happened with ChatGPT [119].

Finally, the improvement of the underlying language models is opening the possibilities toward more and more general agent whose behaviour is controlled through natual language prompts. Thus, eventually, we will drop the adoption of fine-tuning and objective optimisation in favour of prompting as a form of *high-level programming* of the chatbot, as it is (partially) happening with ChatGPT or Bard for example. *Gopher* already showed that with the correct pre-traing and chat prompt the generated dialogues are indistinguishable from those obtained after fine tuning from a human perspective [136].

5.3 Open Problems

Apart from the functional improvements, we expect also the agents to overcome their limitations. In fact, there are still some open technical problems and ethical issues to solve. These issues are inherited from the Seq2Seq language models underlying these chatbots [195].

Seq2Seq chatbot are prone to *knowledge hallucination*. This problem is mostly due to the lack of proper knowledge grounding in favour of weights memorisation(i.e., relying on the knowledge embedded in the model's weights) that characterises the currently developed Seq2Seq bases chatbots (e.g., ChatGPT). Knowledge grounding introduces the possibility of explaining and supporting the generated responses, but increases the system complexity. Moreover, despite there is the possibility of implementing the agent to exploit knowledge sources, there still can be faults in the generated content(the benchmarks in this sense do no report perfect scores [128]). Strictly related to this aspect, is the lack of *sound logical reasoning* that leads these models to generate nonsensical content or leading to the wrong conclusion even if starting from the correct premises (e.g., Gopher at times respondes very confidently with wrong answers [136]).

Given the quality of the generated text and the conversations these chatbots are capable of managing, the prolonged use of these tools, possibly combined with avatars and other embodyment

techniques [122], may result in *overconfidence* towards the agent. This situation can be exploited to manipulate opinion and spread misinformation [195].

These knowledge hallucination and misinformation issues are connected to the *explainability* and *interpretability* of the results. Being deep Neural Networks an *opaque* tool, understanding how and why a specific output is generated is not always easy. Lately, *Human-Computer Interaction*, the are comprising all the technologies that interface directly humans and computers, has become strongly connected to this aspect, leading towards the concept of ***Explainable AI*** (**XAI**) [147]; the idea is to make AI more transparent rather than a black box tool, and subsequently ease its adoption.

Another noteworthy issue is that of *bias* and *fairness* present in the data and the generated content. Being deep Neural Networks based on a frequentist approach to inference, they are prone to overfitting. As a result, if trained on biased/unfair data, these models can possibly produce biased, unfair content such as offensive, discriminative, and hurtful text. Lately, there has been a lot of attention to quantify and remove bias from data and models [20, 45, 61]. Currently, deployed Seq2Seq chatbots (like ChatGPT) use automatic system to detect whether it is generating this kind of content and either avoid it or add a disclaimer to inform the user. However, this detection system are not foolproof.

Finally, given the current computational requirements imposed by these large language models, the access to these technologies for development is limited to very complex and costly cloud deployments or web-based APIs. Eventually, we would expect to be able to get an easier access to the underlying Seq2Seq model to so that everyone could benefit from them and can customise the chatbots to their needs.

APPENDICES

A TRAINING AND EVALUATION

In this section, we explain how Seq2Seq chatbots can be trained and evaluated. As for the previous section, hereafter we refer to models using auto-regressive generation. However, the formulae can easily be extended to other models.

A.1 Training Approaches

To produce models that generate fluent responses, Seq2Seq chatbots must be trained with a large amount of data. The training process can be carried out in multiple steps, from *pre-training* [134, 216] to *fine-tuning* [210], following a so-called *curriculum learning* approach [15]. Finally, following recent trends, we introduce also *objective-based training*, which is a *reinforcement learning*-based fine-tuning.

We talk of curriculum learning because the general idea is to start training the chatbot on large text corpora, even from a different domain, and then iteratively refine (fine-tune) the chatbot, shifting it towards the target domain and behaviour. At each different training step, the complexity of the dialogue task is increased and the samples get closer to the those of the target domain. For example, one can start from pre-training on generic text from books, which account for very large data sets (to learn linguistic structure), do a first fine-tuning on conversations scraped from Twitter (to learn dialogue structure) and finally fine-tune again on empathetic dialogue (to learn empathetic behaviour in conversations). Empirically, this curriculum learning approach was shown to yield better results than training directly on the target data [15].

A.1.1 Pre-training. The pre-training step is fundamental to obtaining a fluent chatbot [1, 145, 216]. The idea is to leverage a large collection of unlabelled text to perform the first training of

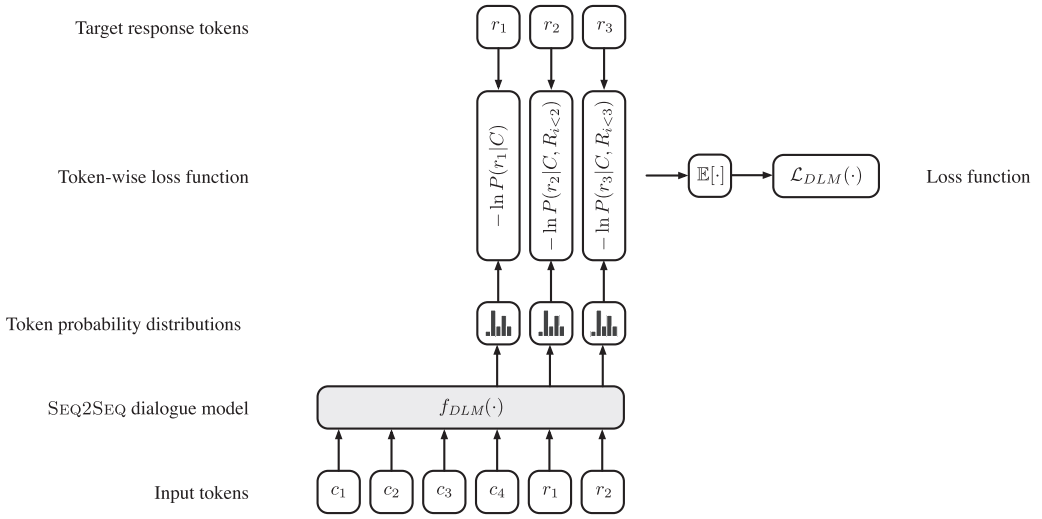


Fig. 25. Visualisation of the loss computation process. The token-wise loss is averaged across the response tokens.

the SEQ2SEQ model. At this training step, the Neural Network learns useful hidden representations and fundamental linguistic structures.

The training data does not necessarily need to be conversations: a generic corpus for language modelling is usually sufficient (e.g., the *Toronto Book Corpus* [222] or *C4* [51]). However, leveraging a large, generic conversation corpus directly yields a usable chatbot, like *Meena* [1] or *BlenderBot* [169] (the latter is then refined via objective optimisation, see Appendix A.1.3). It is also possible to mix these two approaches, performing a first pre-training on generic text data and then a further pre-training on a large, generic conversation corpus, like in the case of *DialoGPT* [216].

Note that such a "large, generic conversation corpus" usually does not contain the conversation typologies that the chatbot will need to learn at the end of the whole training process. So, the goal is to obtain a good initialisation from which it will be simpler to refine the chatbot on the desired (typically smaller) specific conversational corpus.

The language modelling head of the chatbot tries, at each step, to predict the probability distribution of the next token. Thus, the main loss function to minimise is the *negative log-likelihood* of the next token into the sequence, given the preceding ones. In fact, during training, the decoding is guided by the reference response, and thus it is possible to compute the *average negative log-likelihood* loss as reported in Equation (41) and depicted in Figure 25.

$$\mathcal{L}_{DLM}(C, R; \vartheta) = -\frac{1}{|R|}\sum_{i=1}^{|R|} \ln P_{DLM}\left(r_i \middle| C, R_{i'<i}; \vartheta\right), \tag{41}$$

where $C = \langle c_1, \ldots, c_{|C|}\rangle$ is the input context, $R = \langle r_1, \ldots, r_{|R|}\rangle$ is the output response, and ϑ represents the model parameters.

This approach is valid for simple latent hierarchical models, too. For variational latent hierarchical models, the loss function changes to optimising the ***Exponential Lower BOund*** **(ELBO)** [174] (see Equation (42)):

$$\mathcal{L}_{ELBO}(C, R; \vartheta) = -\frac{1}{|R|}\sum_{i=1}^{|R|} \ln P_{DLM}\left(r_i \middle| C, \mathbf{z}, R_{i'<i}; \vartheta_h, \vartheta_{lm}, \vartheta_q\right) + \mathrm{D}_{KL}[q(\mathbf{z}|C, R; \vartheta; \vartheta_h, \vartheta_q)||p(\mathbf{z}|C; \vartheta_h, \vartheta_p)], \tag{42}$$

where C and R are defined as before, $\mathbf{z}$ is the latent variable, sampled from the posterior latent distribution $q(\cdot)$, and $\mathrm{D}_{KL}[q(\cdot)||p(\cdot)]$ is the *Kullbak-Leibler divergence* (or KL divergence[11]) between the prior latent distribution $p(\mathbf{z}|C)$ and the posterior latent distribution $q(\mathbf{z}|C, R)$, defined as in Section 3.2.2. Moreover, ϑ_h, ϑ_{lm}, ϑ_p, and ϑ_q are ,respectively, the parameters of the hidden transformations, the language model head, the prior latent model, and the posterior latent model. Notice that the next token in the response also depends on $\mathbf{z}$, and not only on the context and the previous response tokens.

These models suffer from an issue called *KL vanishing* [218], where the KL-divergence goes to zero cancelling the contribution of the latent. In practice, the variational model degenerates and always predicts the same latent code, making the latent code uninformative and thus useless for the generation process. However, various solutions help prevent this collapse [56, 124, 217, 218, 221]. *Plato* [9, 10] is an example of variational latent hierarchical model pre-trained on conversations scraped from Reddit, it was also fine-tuned on some reference benchmarks (see Appendix A.1.2).

As mentioned before, this first pre-training step requires processing large text collections using a high amount of computational time and power. Thus, starting completely from scratch and doing the entire pre-training is not always viable. So, to cope with this computational power demand, it is possible to rely on pre-trained models made available by large companies (e.g., OpenAI, Google, Microsoft, Facebook). For example, the *Transformers* package from *Hugging Face* [201] gives easy access to many pre-trained Transformer models.

A.1.2 Fine-tuning. Fine-tuning (together with *transfer learning* [210]) has become a fundamental step (especially in NLP) for taking advantage of deep pre-trained models and achieving impressive performances, even on small corpora [46, 134, 135, 137]. This step is part of the curriculum learning process currently used to train chatbots: the model is first trained on a large text corpus (not necessarily conversations, but also books or web scrapes), then it is interatively refined on data samples coming from domains more and more similar to the target one (e.g., corpora for knowledge grounded conversations or empathetic conversations) [86, 99, 168, 202]. In this way, it is possible to take advantage of the initialisation given by the pre-training, which offers a good initial hidden representation encoding the main linguistic features, to achieve good results in an efficient way.

From a practical perspective, the main loss function to optimise for training the network (i.e., the negative log-likelihood of the next token) does not change: only the underlying corpus changes. However, it has become common practice to add further loss functions to improve the fine-tuning process. In the case of chatbots, the language modelling loss function (as defined in Equation (41)) is often mixed with a retrieval one, as in the hybrid architectures described in Section 4.1.1, like *TransferTransfo* [202] or *CAiRE* [99]. During training, the usual negative log-likelihood is paired with a binary *contrastive loss*: the model is presented with samples where the correct response is substituted with a distractor response, following the given context [27, 94, 202]. This approach is used to help chatbots generate better responses [202]. The most straightforward implementation of this approach is to use a separate classifier on top of the decoder model. In such a case, the loss becomes like the one in Equation (43), where $\mathcal{R}_{distractor}$ is the set of distractor responses, α is the hyper-parameter used to control the relative importance of the losses, and $\mathcal{L}_{CLS}(\cdot)$ is the contrastive binary cross-entropy loss to train the retrieval head, defined in Equation (44).

$$\mathcal{L}_{multi\text{-}objective}\left(C, R, \mathcal{R}_{distractor}; \vartheta\right) = \alpha \mathcal{L}_{LM}(C, R) + (1-\alpha)\mathcal{L}_{CLS}\left(C, R, \mathcal{R}_{distractor}\right), \tag{43}$$

[11]The KL divergence is a common way of measuring the distance of a probability distribution from a reference one. In our case $q(\cdot)$ is the reference distribution and $p(\cdot)$ is the distribution to be measured.

$$\mathcal{L}_{CLS}\left(C, R, \mathcal{R}_{distractor}; \vartheta\right) = -\ln P_{CLS}(R \in C_{correct}|C) - \frac{1}{|\mathcal{R}_{distractor}|} \sum_{R' \in \mathcal{R}_{distractor}} \ln P_{CLS}(R' \notin C_{correct}|C) =$$
$$= -\ln P_{CLS}(R \in C_{correct}|C) - \frac{1}{|\mathcal{R}_{distractor}|} \sum_{R' \in \mathcal{R}_{distractor}} \ln\left(1 - P_{CLS}(R' \in C_{correct}|C)\right). \tag{44}$$

Another approach is to leverage *unlikelihood training*, mixing the usual token-wise loss with a contrastive loss [27, 94]. The model is trained to maximise the probability of observing the tokens of the correct response R and minimise the probability of observing the tokens of distractor responses taken from a set of distractors $\mathcal{R}_{distractor}$, as in Equation (45). This second step is done through the *unlikelihood loss* from Equation (46).

$$\mathcal{L}_{contrastive}\left(C, R, \mathcal{R}_{distractor}; \vartheta\right) = \alpha \mathcal{L}_{LM}(C, R) + (1 - \alpha)\mathcal{L}_{UL}\left(C, \mathcal{R}_{distractor}\right), \tag{45}$$

$$\mathcal{L}_{UL}\left(C, \mathcal{R}_{distractor}; \vartheta\right) = \frac{1}{|\mathcal{R}_{distractor}|} \sum_{R' \in \mathcal{R}_{distractor}} -\frac{1}{|R'|} \sum_{i=1}^{|R'|} \ln\left(1 - P_{DLM}\left(r'_i \middle| C, R'_{i'<i}; \vartheta\right)\right). \tag{46}$$

Finally, we want to underline that this fine-tuning step is also useful for introducing additional features to the agent, like conditioning. In general, conditioning affects how the input is presented to the model or which parameters to update during the training iterations (see Section 4.3). Thus, it is helpful to train additional predictive or discriminative parts of the chatbot during fine-tuning. For example, conditioned models, which require the attributes describing the response to generate the answer, would need to train an additional model head to predict such attributes given the context. For example, a model generating responses conditioned on emotion would require an additional module to predict the emotion of the response. Such modules for conditioned language modelling, attribute prediction, and attribute recognition can be trained at once, summing all the losses together, this is the case of *CAiRE* [99] and *EmpTransfo* [211] for empathetic dialogues, they both predict the target emotional status to condition the response generation.

A.1.3 Objective Optimisation. Despite open-domain dialogue not having a clear and well defined objective function to optimise, it is possible to use *reinforcement learning* algorithms [180] to further fine-tune a chatbot to pursue one or more objectives through dialogue [149, 158]. In fact, training a dialogue agent to optimise some objective is more common for task-oriented agents [57].

Most solutions relied on *policy gradient* algorithms, like *REINFORCE* [200], and handcrafted objectives [149, 158, 166, 220]. They are applicable either to base models (defining a *policy* over the tokens composing the response, altering the language model) [93, 166] or to hierarchical models (defining a *policy* over high-level attributes to condition the response generation or on the hidden latent representation) [149, 158, 220]. When used with base models, this kind of approaches may break the generative capabilities of the model and lead to disfluent text, there are possible solutions to deal with these issues [166].

Usually, policy-gradient approaches applied to dialogue generation consider an entire conversation as a sequence of utterances and extract all the avaialble context-response pairs:

$$X = \langle U_1, \ldots, U_i, \ldots, U_{n_X} \rangle \rightarrow \langle (C_1, R_1), \ldots, (C_i, R_i), \ldots, (C_{n_X}, R_{n_X}) \rangle, \tag{47}$$

where $U_i \in \mathcal{V}^{|U_i|}$ is a sequence of tokens representing a turn in the dialogue, $C_i = \langle U_1, \ldots, U_{i-1} \rangle$ is the context associated to the ith turn in the dialogue and $R_i = U_i$ is the ith turn in the dialogue. Each response R_i has an associated reward r_i, so that given the discount factor $\gamma \in [0, 1] \subseteq \mathbb{R}$ of the ***Markov Decision Process*** **(MDP)** [93, 166, 180] associated with the reinforcement learning problem, we can compute the *discounted cumulative future reward* G_i of the response R_i as in

Equation (48)

$$G_i = \sum_{k=0}^{\lfloor (n_X - i)/2 \rfloor} \gamma^k \cdot r_{i+2k}. \tag{48}$$

Note that in some cases, the discounted cumulative future reward is standardised over the dialogue to enforce stability of the policy gradient algorithm. Given a dialogue X, the objective $J_\pi(\cdot)$ to *maximise* to train the dialogue policy is thus defined in Equation (49)

$$J_\pi(X;\vartheta) = -\frac{1}{n_X} \cdot \sum_{i=1}^{n_X} (-G_i \cdot \ln P_{DLM}(R_i|C_i;\vartheta)). \tag{49}$$

To prevent the collapse of the underlying language model, the objective is mixed with the usual language modelling loss, as in Equation (50) [166]

$$J_{mixed}(X;\vartheta) = \alpha J_\pi(X;\vartheta) - (1-\alpha)\mathbb{E}_{(C_i,R_i)\in X}\left[\mathcal{L}_{DLM}(C_i,R_i;\vartheta)\right], \tag{50}$$

with $\alpha \in [0,1] \subseteq \mathbb{R}$ being the hyper-parameter used to control the relative importance of the two objectives

These reinforcement learning-based solutions are particularly suitable for *empathetic chatbots* or *empathetic chatbots*. In fact, as long as a measure of empathy is provided, it is possible to refine the agent to maintain the open-domain dialogue properties while steering the responses towards more empathetic ones. For example, XiaoIce [220] optimises the *expected Conversation-turns Per Session* to be more social and empathetic, which is an handcrafted metric that should correlate with an empathetic behaviour. Another adopted handcrafted feature is the *expected user sentiment*, with the objective of eliciting a positive sentiment in the other person during the dialogue [166]. Instead of handcrafting the objective, some solutions relied on a learnt empathy measure to maximise [163, 164].

Rather then defining an handcrafted objective or learning a specific reward, it is possible to rely on human feedback to improve the agent. *BlenderBot* [169] proposed a solution of continuous improvement where each response can get a feedback with different level of granularity [207]. These feedbacks can be used to learn either a re-ranking system or a reward function to optimise.

ChatGPT [118] represents the current state of the art of open-domain chatbots. It was trained starting from a large language model and then refined through reinforcement learning using a ***Proximal Policy Optimisation*** (**PPO**) algorithm [152]. The model starts from a supervised initialisation and is iteratively updated to optimise a reward function. The reward function is learned by mimicking human ranking on possible responses to a prompt.

A.2 Evaluation Approaches

Two main approaches exist to evaluate generative conversational agents: automatic or human-based [100, 172, 187]. The former is based on automatic metrics that allow a quantitative and objective evaluation of the conversational agent. The latter, instead, is based on subjective human evaluations. Since there is no objective metric capable of capturing all the nuances of an open-domain conversational agent, the task of evaluation is often delegated to humans.

A.2.1 Automatic Evaluation. There are several metrics used to evaluate chatbots. The most popular automatic metric is undoubtedly the *perplexity* (PPL). PPL of generative dialogue models is defined as in Equation (51), where, as usual, C is the context sequence and R is the response sequence.[12] It measures how well a probability distribution predicts a sample. Recent studies found

[12]In this formulation we considered an autoregressive model, but the formula is extensible to any model.

a correlation between PPL and human opinion on different aspects of a conversation [104].

$$\text{PPL}(C, R) = \exp\left(-\frac{1}{|R|}\sum_{i=1}^{|R|}\ln P(r_i|C, R_{i'<i})\right). \tag{51}$$

Other popular metrics are the *next token accuracy* or the F_1*-score*. Note that, despite being defined for the retrieval domain, these metrics also apply to generative models.

Next token accuracy is computed by means of guided decoding. At each step, given the preceding response tokens, the probability distribution of the next token is computed, and the most probable next token is matched against the correct one from the corpus. The accuracy is then the ratio between correct predictions and total response length.

In the case of F_1 score, given a target sequence $R = \langle r_1, \ldots, r_i, \ldots, r_{|R|}\rangle$ and a generated sequence $\widehat{R} = \langle \hat{r}_1, \ldots, \hat{r}_j, \ldots, \hat{r}_{|\widehat{R}|}\rangle$, it is possible to define the *precision* Pr as the ratio of the number of common tokens between R and $\widehat{R}$ and the number of elements in the generated sequence (see Equation (52)). The *recall* Re is defined as the ratio of the number of common tokens between R and $\widehat{R}$ and the number of elements in the target sequence (see Equation (53)). Given these definitions, the F_1 can be computed, as usual, as the harmonic mean of Pr and Re.[13]

$$\text{Pr} = \frac{|\widehat{R} \cap R|}{|R|}, \tag{52}$$

$$\text{Re} = \frac{|\widehat{R} \cap R|}{|\widehat{R}|}, \tag{53}$$

$$\text{F}_1 = 2 \cdot \frac{\text{Pr} \cdot \text{Re}}{\text{Pr} + \text{Re}}. \tag{54}$$

Note that these definitions of Pr, Re, and F_1 consider only *unigrams* (i.e., individual tokens in the sequences). However, we can extend the definitions to consider *n-grams* with $n \geq 1$ (i.e., subsequences of n consecutive tokens), like *bigrams* ($n = 2$), *trigrams* ($n = 3$), and so on. Thus, we can introduce precision, recall and F_1-score over n-grams: $\text{Pr}(\cdot; n)$, $\text{Re}(\cdot; n)$ and $\text{F}_1(\cdot; n)$

Other widely used metrics are BLEU [123], ROUGE [97], and METEOR [8]. These metrics are defined using n-grams comparison.

BLEU, defined in Equation (55), is the *geometric average* of the precision computed with different n-grams (in fact, usually we talk of BLEU$-n$, where n is the maximum considered n-gram). Usually, the elements of the geometric average are weighted uniformly with $w = 1/n$. The BLEU score if often scaled by a brevity penalty defined in Equation (56).

$$\text{BLEU}(R, \widehat{R}; n) = \text{brevity}-\text{penalty}(R, \widehat{R}) \cdot \prod_{n'=1}^{n} \text{Pr}(R, \widehat{R}; n)^{\frac{1}{n}}, \tag{55}$$

$$\text{brevity}-\text{penalty}(R, \widehat{R}) = \begin{cases} 1 & |\widehat{R}| > |R| \\ \exp\left(1 - \frac{|R|}{|\widehat{R}|}\right) & |\widehat{R}| \leq |R| \end{cases}. \tag{56}$$

ROUGE is a set of metrics: ROUGE$-L$, ROUGE$-S$, and other weighted variants. All these variants are focused on the computation of the recall. ROUGE$-n$ is the n-gram recall. ROUGE$-L$ is recall computed as the ratio of the ***Longest Common Subsequence*** **(LCS)** of R and $\widehat{R}$ and the length of the reference response, see Equation (57). ROUGE$-S$ considers the *skip-bigram* co-occurence statistics, where a skip-bigram is any pair of unigrams in their sentence order; see Equation (58), where the function $\text{skip}(\cdot; n)$ maps the input sequence to its skip-n-grams.

$$\text{ROUGE}-L(R, \widehat{R}) = \frac{\text{LCS}(R, \widehat{R})}{|\widehat{R}|}, \tag{57}$$

[13] Often, when computing these metrics, articles and other similar *stopwords* are removed.

$$\text{ROUGE}-S(R, \widehat{R}) = \frac{|\text{skip}(R; 2) \cap \text{skip}(\widehat{R}; 2)|}{|\text{skip}(\widehat{R}; 2)|}. \tag{58}$$

METEOR relies on alignments, that are mappings between the reference output sequence and the generated output sequence (we provided an example of mapping in Figure 13).

The METEOR score, defined in Equation (59), is a scaled, weighted harmonic mean of precision and recall. The F−mean$(\cdot)$ is the harmonic mean of precision and recall where recall is weighted nine times more than precision, see Equation (60). The scale is computed from the *chunks penalty* defined in Equation (61), where the chunk function computes the fewest possible overlapping chunks between the target response and generated response. A chunk is a set of unigrams that are consecutive in the target output and the generated output.

$$\text{METEOR}(R, \widehat{R}) = (1 - \text{chunks}-\text{penalty}(R, \widehat{R})) \cdot \text{F}-\text{mean}(R, \widehat{R}), \tag{59}$$

$$\text{F}-\text{mean}(R, \widehat{R}) = \frac{10 \cdot \text{Pr}(R, \widehat{R}) \cdot \text{Re}(R, \widehat{R})}{\text{Pr}(R, \widehat{R}) + 9 \cdot \text{Re}(R, \widehat{R})}, \tag{60}$$

$$\text{chunks}-\text{penalty}(R, \widehat{R}) = \frac{1}{2} \cdot \left(\frac{|\text{chunk}(R, \widehat{R})|}{|R \cap \widehat{R}|} \right)^3. \tag{61}$$

These last three metrics come from other areas of NLP (machine translation and automatic summarisation). Similarly to F_1-*score* , such metrics try to evaluate the generated sequence by means of its overlap with a reference sequence. Although useful, these metrics not always correlate well with human judgement, and hence their use can be counterproductive (for example the results of the human evaluation in the *ConvAI2* challenge reported as winner a model that performed poorly on overlap metrics [49]). In cases where the objective is to generate diverse responses that may steer away from the reference ones, the distinct−n metric can be useful to evaluate the generated responses independently of the target responses [91]. distinct−n is defined as the ratio between the distinct n-grams in the generated response and the length (in tokens) of the generated response, see Equation (62) ($n-\text{gram}(\cdot; n)$ maps the input sequence of tokens to its sequence of n-grams). Usually the metric is computed with $n \in \{1, 2, 3\}$.

$$\text{distinct}-n(\widehat{R}; n) = \frac{|n-\text{gram}(\widehat{R}; n)|}{|\widehat{R}|}. \tag{62}$$

Alternatively to these metrics, a recent trend consists in using *learned metrics* [58, 104, 181, 187], where a model is trained on a corpus of human ratings of conversations with dialogue agents. Provided a sufficiently accurate model, it should be possible to have a robust estimate of human evaluations without involving any human in the evaluation loop. Alternatively, it is possible to directly use the raw probabilities of large language models to evaluate dialogue quality automatically. In fact, the PPL of these models showed good correlation with human ratings on some dialogue aspects [116, 178].

A.2.2 Human Evaluation. Usually, human raters are asked to rate specific aspects or compare available responses. The chatbot is then evaluated, for example, on the raters' preferences or based on the number of times the chatbot performed not worse than the ground truth or other models. Evaluated aspects include [1, 91, 140, 154, 181, 214]:

— *General quality*: whether the response is to be considered good;
— *Ease of answer*: whether the response would be easy to respond to;
— *Fluency*: whether the language seems accurate, or the response can be understood;
— *Relevance*: whether the responses seem appropriate to the conversation;

— *Naturalness*: whether the response maintains the natural conversation flow;
— *Consistency*: whether responses contain contradictory information with respect to the context;
— *Diversity*: whether the responses show lexical diversity;
— *Engagingness*: whether the conversation is interesting disregarding fluency;
— *Sensibilness*: whether the response, given the context, makes sense;
— *Sensitivity*: whether the response is specific given the context.

In some cases, open-domain agents are also evaluated on *empathy* [140, 166], to understand whether the responses show understanding of the feelings of the other person. Notice that the raters can evaluate individual responses or can compare alternative responses.

Usually, human raters provide different evaluations. To help reconcile such evaluations, and understand whether the whole evaluation process was successful, an *agreement coefficient* among raters is usually calculated. The *Kappa coefficient* [48] is a popular choice, as it considers the possibility of the agreement occurring by chance.

Many formulas exist to calculate the Kappa (Cohen's K, Scott's pi, Fleiss' kappa, Randolph's kfree [139], etc.) but the basic idea is given by Equation (63):

$$\text{Kappa} = 1 - \frac{1 - p_o}{1 - p_e}, \tag{63}$$

where p_o is the observed agreement among raters, and p_e is the expected probability of chance agreement. Kappa=1 means perfect agreement, while Kappa = 0 means chance agreement. Kappa can be negative if there is no relationship between the ratings or the raters tend to give totally differing ratings. Note, however, that Kappa can be affected by bias and other issues and thus its results should be evaluated with care [48].

Human evaluation can be carried out in two ways: either in a static manner, where the human is given samples composed of context taken from a corpus and response(s) to evaluate, or in an interactive manner, where the human carries out a conversation with an agent, in real-time.

Static. Static human evaluation is the most straightforward approach. The idea is to start from fixed conversation contexts, like those available in the training set, and generate a response. This approach can be applied to compare multiple models. Human raters are given the context, followed by the generated response(s) and the ground truth response, and are asked to evaluate them. It is possible to adopt a so-called *mechanical Turk* of crowdsourced workers [49, 158] to have a larger (but less controllable) pool of raters, or rely on a more restricted (but possibly better) set of expert raters (e.g., three or five) [91–93, 166] to evaluate the models.

Interactive. Differently, from static evaluation, the interactive approach poses some non-trivial challenges. First of all, the chatbot should be responsive in real-time. Given that they need to mimic human behaviour, a high response time may harm the engagingness. Secondly, it requires deploying the agent and hence developing a user interface, or at least leveraging some available APIs from social media (e.g., Facebook Messenger[14] [220]) or messaging services (e.g., Telegram[15] [49]). It this setting, the human can directly interact with the agent, making the evaluation more entertaining. This kind of approach can be helpful in collecting real-time feedback from the users. Such feedback can be leveraged to update the agent either at run-time or after the conversation.

[14]https://www.facebook.com.
[15]https://telegram.org.

B CORPORA AND COMPETITIONS

In this section, we describe the main corpora to train and evaluate Seq2Seq chatbots [157], and the most interesting competitions related to these models. Sometimes, the competitions revolve around one of the available corpora.

B.1 Corpora

Dialogue corpora to train generative chatbots can be of many kinds. We distinguish among three main groups: *pre-training corpora*, *fine-tuning corpora* and *benchmarks or collections*.

B.1.1 Pre-training. Large conversational corpora obtained scraping social media (like Twitter) and forums (like Reddit) are often used to perform pre-training of conversational agents. The advantage of these corpora is that, due to their size –often these corpora include from hundreds of thousands to millions of conversation samples– they provide the Seq2Seq model with basic conversation capabilities that can be easily refined on more specific corpora that usually provide way fewer samples. The disadvantage is the lack of control over the content and the low quality of language in the conversations [157]. It is possible to train agents with impressive linguistic capabilities, leveraging such low-quality scraped conversations, but the amount of required data is huge (for example, Google used more than 300 GB of text to train Meena [1]). In the following, we briefly describe some of such corpora.

The *Ubuntu Corpus* [105] was extracted from the *Ubuntu chat logs*.[16] It is an extensive collection of multi-turn dialogues for neural conversational agents. All conversations are between two human participants.

The *Cornell Movie-Dialogs Corpus* [40] was built from raw movie scripts and covers many topics and writing styles. Unlike other large corpora, it was manually curated and contains metadata about the speaking character and the movie.

Reddit[17] is a website containing social news, web content ratings, and discussions. Due to the conversational structure of the posts, it has been scraped to extract dialogue corpora [77, 216]. To avoid directly scraping the original website, a third party dump of Reddit is published on the *pushift.io*[18] platform. Extracted conversations cover many topics, which reflect the ones of the *sub-reddit* they belong to. Conversations are multi-turn and multi-writer. Moreover, in some cases, it is possible to mine additional information without manual annotations, like the writer's persona [212].

Twitter[19] was one of the first social media to have been employed for these kinds of tasks, due to the public availability of its content [1, 155, 175], and is still occasionally used to build corpora for training the agents. Conversations are built scraping the answers under posts. As for previous corpora, the conversations cover different topics and are multi-turn and multi-writer.

B.1.2 Fine-tuning. Apart from the extensive conversational collections used for pre-training, many corpora exist to train open-domain dialogue agents. Unlike the corpora mentioned above, those for fine-tuning have fewer samples, are manually curated, and are rich in additional information, like labels and metadata. Labels may include emotion, topic, and dialogue acts, while metadata may include information to ground the conversation in:

— speakers' persona, the *persona grounding* is a set of short sentences describing the speaker's profile.

[16] https://irclogs.ubuntu.com.
[17] https://www.reddit.com.
[18] https://pushshift.io.
[19] https://twitter.com.

— knowledge, with *knowledge grounding document* we refer to a document chunk, manually selected to act as reference knowledge for providing grounding to a given response;
— situation, the *situation grounding* is provided as a short description of the situation discussed in the dialogue or that is happening while the dialogue is taking place;
— images, the *image grounding* is provided through one or more pictures that are object of the conversation (or at least mentioned in the conversation).

Early corpora of this kind, like *CallHome* [31], *CallFriend* [30] and *Switchboard* [60], were extracted from recorded telephone conversations. The Switchboard corpus also presents dialogue acts, leveraging the DAMSL notation [4].

A commonly found label, especially in recent corpora, describes the emotion. In fact, thanks to the rising interest in affective computing [129], many corpora have been labelled to include the speaker's emotional status, often employing a categorical representation reflecting Elman's taxonomy [53] (in some cases, the Elman's six basic labels are extended to have a better granularity). Some of these corpora also label the speaker's sentiment (i.e., positive or negative polarity).

Emotion labelled corpora include *SEMAINE* [109] and *IEMOCAP* [26], which provide video and audio recordings of the enacted conversations. IEMOCAP also includes emotion labels with a continuous notation [39]. Differently, the *DailyDialogues* [95] corpus was built crawling conversations from English learning websites and includes labels about emotion, dialogue act, and topic. The *EmpatheticDialogues* [140] contains simulated empathetic dialogues grounded in a situation presented through a descriptive sentence associated with the dialogue (this data set was used to train the empathetic chatbot *CAiRE* [99]). The *MELD* corpus [130] (previously known as *EmoLines* [75]) presents multi-speaker dialogues extracted from the transcripts of the "Friends" TV show, it also presents sentiment labels. The *EDOS* [197] corpus was recently introduced, it is the most extensive available collection of this kind: it contains around a million dialogues extracted from the *OpenSubtitles*[20] data base. Each turn is labelled with emotion or intent information.

Some corpora present specific kinds of dialogues. A relevant example is given by dialogues where one of the speakers tries to explain something to the other. In these corpora, it is possible to observe clarification questions that are hardly used by open-domain chatbots. Corpora of this kind include the *Teacher-Student Chatroom* [28, 29], where teachers answer students' questions, *ELI-5* [55], extracted from a subreddit to provide simplified explanations about any topic, and the *HCRC Map Task* [182] where multiple speakers try to provide someone with directions to reach some place.

In the context of *mental healthcare*, counselling and psychotherapy represent two task-oriented dialogue problems that are actually approached with open-domain dialogue techniques. In fact, counselling or therapy sessions are actually open domain dialogues. There exists some data sets that are . *Counseling and Psychotherapy Transcripts: Volume II* [115] and *HOPE* [107] (now extended to *MEMO* corpus [176]) are transcription of sessions. Moreover, HOPE contains also labels with dialogue acts associated to each utterance, and its extension MEMO contains labels with the status of the session and the summary of the interaction. Additionally, there exists also the *Counsel Chat* [17] and the Epitome [164] corpora. These two corpora are composed only of message-response pairs instead of entire conversations. Epitome is labelled with the communication level of different empathy mechanisms.

Certain corpora present some kind of grounding. It is the case of corpora for persona grounding, which are helpful to ensure agent's consistency. Persona groundings can be provided for the agent, the user or both. Having access to both persona descriptions, improves the agent's dialogue

[20] https://www.opensubtitles.org/.

capabilities [92]. The *Persona-Chat* [214] corpus contains simulated conversations preceded by short sentences describing participants' personas. The corpus also provides variants of the persona descriptions, and contrastive samples for discriminative training and evaluation (this chatbot was used to train *TransferTransfo* [202] in the *ConvAI-2* challenge [49] and to do the first fine-tuning of *CAiRE* [99]). The *LIGHT* [184] corpus, instead, has dialogues grounded in a text adventure game. It also provides information about the actions performed within the game (situation grounding). More recently the *Multi-Session Chat* [206] was released, this corpus uses persona grounding and was specifically realised to learn managing long conversations.

Other forms of grounding include knowledge. Some corpora provide this grounding through document exchanges that are associated with dialogue turns. It is the case of the *Wizard-of-Wikipedia* [50], *Topical-Chat* [64], and *Wizard-of-Internet* [86] corpora. In all cases the speakers refer to external knowledge sources (Wikipedia for the first corpus, many sources for the other two) where text chunks are extracted and added to the conversation turns, as metadata. In the first two corpora, the topic is explicitly associated with the conversation.

Finally, it is worth to mention image grounded corpora, like *Image Chat* [167] and *IGN* [114]. Due to the emerging presence of chatbots in social media, and given the relevance that images play in such contexts, providing chatbots with the capability of handling visual contents is essential. Both corpora are taken from conversations about a given image: Image Chat provides also persona information, IGC is focused on question-answer dialogues.

As a conclusive remark, it is important to point out that the corpora used for fine-tuning are often curated to avoid unsafe and biased content. In some cases, for example, when pre-training on large corpora crawled from the web, the resulting corpus may contain harmful examples. To cope with this issue some filtering techniques have been suggested [205, 216]. These techniques aim at filtering the corpus to remove unsafe, biased, or negative utterances.

B.1.3 Benchmarks and Collections. Lately, there has been an emerging interest in building benchmarks for conversational agents. The idea behind these benchmarks is to collect multiple corpora covering different aspects and dialogue styles, and combine them to test automatically all these capabilities. These benchmarks offer both training and evaluation data. In the following, some of them are presented.

The *DodecaDialogue* [168] covers 12 different corpora that are used both for pre-training and fine-tuning. The fine-tuning ones allow for grounding on persona, knowledge, situation, and images. It is mainly though for generative agents since it evaluates the agent generative abilities in many different contexts. In fact, the baseline models trained on these corpora have been evaluated on PPL, F_1-score, BLEU and ROUGE.

The *Silicone* [33] benchmark, instead, is mostly for sequence labelling rather than generation. It is composed of eight different corpora, including one for task-oriented conversations. The labels of the corpora provide dialogue acts and affect (either emotion or sentiment). Despite being for sequence labelling, this benchmark can be employed to evaluate generative agents in the same way as DodecaDialogue.

The *KILT* [128] benchmark is a resource for *knowledge intensive language tasks*. This benchmark is composed of 11 different data sets targeting five tasks. Apart from the dialogue task (provided by the Wizard-of-Wikipedia data set), KILT includes open domain question answering, slot filling, entity linking and fact checking. This benchmark is useful to improve knowledge grounded agents and to learn to provide support to factual responses.

The *BlendedSkillTalk* [173] targets a chatbot ability to manage different tasks within a conversation. The considered tasks are to talk about oneself as well as getting to know the conversation partner, displaying empathetic behaviour and being knowledgeable. BlendedSkillTalk is built

starting from pre-exisiting data sets (namely Persona-Chat, EmpatheticDialogue and Wizard-of-Wikipedia) that are used as prompts for human workers to generate new dialogues. These dialogues are labelled to help the agent understand whether an utterance in the dialogue requires external knowledge, requires personal knowledge, describes a personal situation, or is a display of empathy. This benchmark permits to evaluate the grounding skills of the agent as well as its ability to understand when and which grounding is necessary.

B.2 Competitions

Competitions for chatbots are thought to promote sharing the latest results in the design of conversational systems. They cover many kinds of approaches for open-domain models and sometimes involve multiple interaction modalities.

The *Loebner Prize Competition* (Loebner prize for AI) [131] was the oldest competitions for chatbots. It is currently discontinued and was based on the Turing Test for machine intelligence. The latest registered winner (in 2018) was the *Mitsuku* chatbot (now *Kuki AI*) [203], which is based on the ***AI Markup Language (AIML)*** for its core functions [190].

The ***Dialog State Tracking Challenge*** (**DSTC**) [47, 67, 73, 82] was first organized in 2013 and reached the *10th edition*.[21] It is mainly thought for task-oriented systems. However, nowadays, this challenge is divided into multiple tracks, also involving open-domain conversational agents.

The *NTCIR Short Text Conversation Task* [161, 162, 213, 215] started as a retrieval task for Chinese and Japanese chatbots, but it was extended to include generative models, from the second edition. It is mainly designed for "post-comments" kind of conversations found in social media like *Weibo*[22] or Twitter. In the latest edition, the organisers also introduced a task for emotion grounded conversations.

The ***Conversational AI Challenge*** (**ConvAI**) [3, 25, 49] is a series of competitions for open-domain agents. Each edition presents a new challenge (e.g., "clarifying questions"). The second edition is particularly famous for the results obtained by transfer-learning and fine-tuned models[202], in the persona grounded conversations (on Persona-Chat [214]). Such transfer-learning based models have become a prevalent approach for neural chatbots, as explained in Appendix A.1.2.

The *Alexa Prize* [138] is a challenge organised by Amazon to improve the Alexa virtual assistant. The idea is to give chit-chat capabilities to Alexa to make it more entertaining. The prize currently proposes three challenges, including one for social chatbots, which is specifically thought for open-domain conversational agents.

REFERENCES

[1] Daniel Adiwardana, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a human-like open-domain chatbot. arXiv:2001.09977. Retrieved from https://arxiv.org/abs/2001.09977.

[2] Manish Agnihotri, Pooja Rao S. B., Dinesh Babu Jayagopi, Sushranth Hebbar, Sowmya Rasipuram, Anutosh Maitra, and Shubhashis Sengupta. 2020. Towards generating topic-driven and affective responses to assist mental wellness. In *Pattern Recognition. ICPR Int. Workshops and Challenges - Virtual Event, January 10–15, 2021, Proceedings, Part II.* Alberto Del Bimbo, Rita Cucchiara, Stan Sclaroff, Giovanni Maria Farinella, Tao Mei, Marco Bertini, Hugo Jair Escalante, and Roberto Vezzani (Eds.), Lecture Notes in Computer Science, Vol. 12662, Springer, 129–143.

[3] Mohammad Aliannejadi, Julia Kiseleva, Aleksandr Chuklin, Jeff Dalton, and Mikhail S. Burtsev. 2020. ConvAI3: Generating clarifying questions for open-domain dialogue systems (ClariQ). arXiv:2009.11352. Retrieved from https://arxiv.org/abs/2009.11352.

[4] James Allen and Mark Core. 1997. Draft of DAMSL: Dialog act markup in several layers. https://www.cs.rochester.edu/research/cisd/resources/damsl/RevisedManual/.

[21] https://dstc10.dstc.community.

[22] https://weibo.com/.

[5] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *Proceedings of the 5th International Conference on Learning Representations*. OpenReview.net.

[6] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. arXiv:1607.06450. Retrieved from https://arxiv.org/abs/1607.06450.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd Int. Conf. on Learning Representations*. Yoshua Bengio and Yann LeCun (Eds.).

[8] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005*. Jade Goldstein, Alon Lavie, Chin-Yew Lin, and Clare R. Voss (Eds.), Association for Computational Linguistics, 65–72.

[9] Siqi Bao, Huang He, Fan Wang, Hua Wu, and Haifeng Wang. 2020. PLATO: Pre-trained dialogue generation model with discrete latent variable. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.), Association for Computational Linguistics, 85–96.

[10] Siqi Bao, Huang He, Fan Wang, Hua Wu, Haifeng Wang, Wenquan Wu, Zhen Guo, Zhibin Liu, and Xinchao Xu. 2021. PLATO-2: Towards building an open-domain chatbot via curriculum learning. In *Proceedings of the Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1–6, 2021 (Findings of ACL, Vol. ACL/IJCNLP 2021)*. Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.), Association for Computational Linguistics, 2513–2525.

[11] Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. In *Proceedings of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3–7, 2019*. Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.), Association for Computational Linguistics, 1538–1548.

[12] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22–27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*. The Association for Computer Linguistics, 238–247.

[13] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Proceedings of the Advances in Neural Information Processing Systems 13*. Todd K. Leen, Thomas G. Dietterich, and Volker Tresp (Eds.), MIT Press, 932–938.

[14] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3, 2 (2003), 1137–1155. https://www.jmlr.org/papers/v3/bengio03a.html.

[15] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. Andrea Pohoreckyj Danyluk, Léon Bottou, and Michael L. Littman (Eds.), ACM, 41–48.

[16] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. *Neural Probabilistic Language Models*. Springer, Berlin, 137–186.

[17] Nicolas Bertagnolli. 2020. Counsel Chat: Bootstrapping High-Quality Therapy Data. Retrieved from https://towardsdatascience.com/counsel-chat-bootstrapping-high-quality-therapy-data-971b419f33da.

[18] Christopher M. Bishop. 2007. *Pattern Recognition and Machine Learning, 5th Edition*. Springer. Retrieved from https://www.worldcat.org/oclc/71008143.

[19] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146. https://aclanthology.org/Q17-1010/.

[20] Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. arXiv:1607.06520. Retrieved from http://arxiv.org/abs/1607.06520.

[21] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conf. on Computational Natural Language Learning*. Yoav Goldberg and Stefan Riezler (Eds.), ACL, 10–21.

[22] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conf. on Neural Information Processing Systems 2020*. Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

[23] Pawel Budzianowski and Ivan Vulic. 2019. Hello, It's GPT-2 - how can I help you? Towards the use of pretrained language models for task-oriented dialogue systems. In *Proceedings of the 3rd Workshop on Neural Generation and Translation@EMNLP-IJCNLP 2019*, Alexandra Birch, Andrew M. Finch, Hiroaki Hayashi, Ioannis Konstas, Thang Luong, Graham Neubig, Yusuke Oda, and Katsuhito Sudoh (Eds.), Association for Computational Linguistics, 15–22.

[24] Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. MultiWOZ - A large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conf. on Empirical Methods in Natural Language Processing*. Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.), Association for Computational Linguistics, 5016–5026.

[25] Mikhail Burtsev, Varvara Logacheva, Valentin Malykh, Iulian Vlad Serban, Ryan Lowe, Shrimai Prabhumoye, Alan W. Black, Alexander Rudnicky, and Yoshua Bengio. 2018. The first conversational intelligence challenge. In *Proceedings of the NIPS'17 Competition: Building Intelligent Systems*. Sergio Escalera and Markus Weimer (Eds.), Springer International Publishing, Cham, 25–46.

[26] Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N. Chang, Sungbok Lee, and Shrikanth S. Narayanan. 2008. IEMOCAP: Interactive emotional dyadic motion capture database. *Lang. Resour. Evaluation* 42, 4 (2008), 335–359.

[27] Hengyi Cai, Hongshen Chen, Yonghao Song, Zhuoye Ding, Yongjun Bao, Weipeng Yan, and Xiaofang Zhao. 2020. Group-wise contrastive learning for neural dialogue generation. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16–20 November 2020 (Findings of ACL, Vol. EMNLP 2020)*. Trevor Cohn, Yulan He, and Yang Liu (Eds.), Association for Computational Linguistics, 793–802.

[28] Andrew Caines, Helen Yannakoudakis, Helen Allen, Pascual Pérez-Paredes, Bill Byrne, and Paula Buttery. 2022. The teacher-student chatroom corpus version 2: More lessons, new annotation, automatic detection of sequence shifts. In *Proceedings of the 11th Workshop on NLP for Computer Assisted Language Learning*. LiU Electronic Press, Louvain-la-Neuve, Belgium, 23–35. Retrieved from https://aclanthology.org/2022.nlp4call-1.3.

[29] Andrew Caines, Helen Yannakoudakis, Helena Edmondson, Helen Allen, Pascual Pérez-Paredes, Bill Byrne, and Paula Buttery. 2020. The teacher-student chatroom corpus. arXiv:2011.07109. Retrieved from https://arxiv.org/abs/2011.07109.

[30] Alexandra Canavan, David Graff, and George Zipperlen. 1996. CALLFRIEND American English-non-southern Dialect. https://doi.org/10.35111/d37s-c536

[31] Alexandra Canavan, David Graff, and George Zipperlen. 1997. CALLHOME American English Speech. https://doi.org/10.35111/exq3-x930

[32] Rollo Carpenter. 1997. Cleverbot. https://www.cleverbot.com/app.

[33] Emile Chapuis, Pierre Colombo, Matteo Manica, Matthieu Labeau, and Chloé Clavel. 2020. Hierarchical pre-training for sequence labelling in spoken dialog. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16–20 November 2020 (Findings of ACL, Vol. EMNLP 2020)*. Trevor Cohn, Yulan He, and Yang Liu (Eds.), Association for Computational Linguistics, 2636–2648.

[34] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of the SSST@EMNLP 2014, 8th Workshop on Syntax, Semantics and Structure in Statistical Translation*. Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi (Eds.), Association for Computational Linguistics, 103–111.

[35] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555. Retrieved from https://arxiv.org/abs/1412.3555.

[36] Ronan Collobert and Jason Weston. 2007. Fast semantic extraction using a novel neural network architecture. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*. John A. Carroll, Antal van den Bosch, and Annie Zaenen (Eds.), The Association for Computational Linguistics.

[37] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the Machine Learning, Proceedings of the 25th Int. Conf. (ICML 2008), Helsinki, Finland, June 5–9, 2008 (ACM Int. Conf. Proceeding Series, Vol. 307)*. William W. Cohen, Andrew McCallum, and Sam T. Roweis (Eds.), ACM, 160–167.

[38] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12 (2011), 2493–2537. https://jmlr.org/papers/v12/collobert11a.html.

[39] Roddy Cowie and Randolph R. Cornelius. 2003. Describing the emotional states that are expressed in speech. *Speech Communication*. 40, 1–2 (2003), 5–32.

[40] Cristian Danescu-Niculescu-Mizil and Lillian Lee. 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*. Frank Keller and David Reitter (Eds.), Association for Computational Linguistics, 76–87.

[41] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *Proceedings of the 8th Int. Conf. on Learning Representations*. OpenReview.net.

[42] Yann N. Dauphin, Harm de Vries, Junyoung Chung, and Yoshua Bengio. 2015. RMSProp and equilibrated adaptive learning rates for non-convex optimization. arXiv:1502.04390. Retrieved from http://arxiv.org/abs/1502.04390.

[43] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41, 6 (1990), 391–407.

[44] Scott C. Deerwester, Susan T. Dumais, George W. Furnas, Richard A. Harshman, Thomas K. Landauer, Karen E. Lochbaum, and Lynn A. Streeter. 1989. Computer Information Retrieval using Latent Semantic Structure. https://patents.google.com/patent/US4839853A/en.

[45] Pieter Delobelle, Ewoenam Tokpo, Toon Calders, and Bettina Berendt. 2022. Measuring fairness with biased rulers: A comparative study on bias metrics for pre-trained language models. In *Proceedings of the 2022 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 1693–1706. DOI: https://doi.org/10.18653/v1/2022.naacl-main.122

[46] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Jill Burstein, Christy Doran, and Thamar Solorio (Eds.), Association for Computational Linguistics, 4171–4186.

[47] Luis Fernando D'Haro, Koichiro Yoshino, Chiori Hori, Tim K. Marks, Lazaros Polymenakos, Jonathan K. Kummerfeld, Michel Galley, and Xiang Gao. 2020. Overview of the seventh dialog system technology challenge: DSTC7. *Computer Speech & Language* 62 (2020), 101068. https://www.sciencedirect.com/science/article/abs/pii/S0885230820300012.

[48] Barbara Di Eugenio and Michael Glass. 2004. The kappa statistic: A second look. *Computational Linguistics* 30, 1 (2004), 95–101. DOI: https://doi.org/10.1162/089120104773633402

[49] Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander H. Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, Shrimai Prabhumoye, Alan W. Black, Alexander I. Rudnicky, Jason Williams, Joelle Pineau, Mikhail S. Burtsev, and Jason Weston. 2019. The second conversational intelligence challenge (ConvAI2). arXiv:1902.00098. Retrieved from https://arxiv.org/abs/1902.00098.

[50] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *Proceedings of the 7th Int. Conf. on Learning Representations*. OpenReview.net.

[51] Jesse Dodge, Maarten Sap, Ana Marasovic, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. In *Proceedings of the 2021 Conf. on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7–11 November, 2021*. Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.), Association for Computational Linguistics, 1286–1305.

[52] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. 1988. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI Conf. on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, 281–285.

[53] Paul Ekman. 1992. Facial expressions of emotion: New findings, new questions. *Psychological Science* 3, 1 (1992), 34–38 (5 pages).

[54] Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science* 14, 2 (1990), 179–211.

[55] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Conf. of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Anna Korhonen, David R. Traum, and Lluís Màrquez (Eds.), Association for Computational Linguistics, 3558–3567.

[56] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. 2019. Cyclical annealing schedule: A simple approach to mitigating KL vanishing. In *Proceedings of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Jill Burstein, Christy Doran, and Thamar Solorio (Eds.), Association for Computational Linguistics, 240–250.

[57] Jianfeng Gao, Michel Galley, and Lihong Li. 2019. Neural approaches to conversational AI. *Found. Trends Inf. Retr.* 13, 2–3 (2019), 127–298.

[58] Sarik Ghazarian, Johnny Tian-Zheng Wei, Aram Galstyan, and Nanyun Peng. 2019. Better automatic evaluation of open-domain dialogue systems with contextualized embeddings. arXiv:1904.10635. Retrieved from https://arxiv.org/abs/1904.10635.

[59] Emer Gilmartin, Christian Saam, Carl Vogel, Nick Campbell, and Vincent Wade. 2018. Just talking - modelling casual conversation. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. Kazunori Komatani, Diane J. Litman, Kai Yu, Lawrence Cavedon, Mikio Nakano, and Alex Papangelis (Eds.), Association for Computational Linguistics, 51–59.

[60] John J. Godfrey, Edward Holliman, and Jane McDaniel. 1992. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of the 1992 IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*. IEEE Computer Society, 517–520.

[61] Hila Gonen and Yoav Goldberg. 2019. Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them. In *Proceedings of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Jill Burstein, Christy Doran, and Thamar Solorio (Eds.), Association for Computational Linguistics, 609–614. DOI: https://doi.org/10.18653/v1/n19-1061

[62] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. 2016. *Deep Learning*. MIT Press. Retrieved from http://www.deeplearningbook.org/.

[63] Google. 2023. Introducing Bard. Retrieved from https://bard.google.com.

[64] Karthik Gopalakrishnan, Behnam Hedayatnia, Qinglang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tür. 2019. Topical-chat: Towards knowledge-grounded open-domain conversations. In *Proceedings of the Interspeech 2019, 20th Annual Conf. of the Int. Speech Communication Association*. Gernot Kubin and Zdravko Kacic (Eds.), ISCA, 1891–1895.

[65] Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. arXiv:1711.02281. Retrieved from https://arxiv.org/abs/1711.02281.

[66] Xiaodong Gu, Kang Min Yoo, and Jung-Woo Ha. 2021. DialogBERT: Discourse-aware response generation via learning to recover and rank utterances. In *Proceedings of the 35th AAAI Conf. on Artificial Intelligence, AAAI 2021, 33rd Conf. on Innovative Applications of Artificial Intelligence, IAAI 2021, The 11th Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 12911–12919.

[67] R. Chulaka Gunasekara, Seokhwan Kim, Luis Fernando D'Haro, Abhinav Rastogi, Yun-Nung Chen, Mihail Eric, Behnam Hedayatnia, Karthik Gopalakrishnan, Yang Liu, Chao-Wei Huang, Dilek Hakkani-Tür, Jinchao Li, Qi Zhu, Lingxiao Luo, Lars Liden, Kaili Huang, Shahin Shayandeh, Runze Liang, Baolin Peng, Zheng Zhang, Swadheen Shukla, Minlie Huang, Jianfeng Gao, Shikib Mehri, Yulan Feng, Carla Gordon, Seyed Hossein Alavi, David R. Traum, Maxine Eskénazi, Ahmad Beirami, Eunjoon Cho, Paul A. Crook, Ankita De, Alborz Geramifard, Satwik Kottur, Seungwhan Moon, Shivani Poddar, and Rajen Subba. 2020. Overview of the ninth dialog system technology challenge: DSTC9. arXiv:2011.06486. Retrieved from https://arxiv.org/abs/2011.06486.

[68] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conf. on Computer Vision and Pattern Recognition*. IEEE Computer Society, 770–778.

[69] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *Proceedings of the 9th Int. Conf. on Learning Representations*. OpenReview.net. https://openreview.net/forum?id=XPZIaotutsD.

[70] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[71] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models. arXiv:2203.15556. Retrieved from https://arxiv.org/abs/2203.15556.

[72] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *Proceedings of the 8th Int. Conf. on Learning Representations*. OpenReview.net.

[73] Chiori Hori and Takaaki Hori. 2017. End-to-end conversation modeling track in DSTC6. arXiv:1706.07440. Retrieved from https://arxiv.org/abs/1706.07440.

[74] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th Int. Conf. on Machine Learning*. Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), PMLR, 2790–2799.

[75] Chao-Chun Hsu, Sheng-Yeh Chen, Chuan-Chun Kuo, Ting-Hao Huang, and Lun-Wei Ku. 2018. EmotionLines: An emotion corpus of multi-party conversations. In *Proceedings of the 11th Int. Conf. on Language Resources and Evaluation*. European Language Resources Association (ELRA), Miyazaki, Japan.

[76] Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. 2020. Challenges in building intelligent open-domain dialog systems. *ACM Transactions on Information Systems* 38, 3 (2020), 21:1–21:32.

[77] Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2020. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *Proceedings of the 8th Int. Conf. on Learning Representations*. OpenReview.net.

[78] Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proceedings of the 5th Int. Conf. on Learning Representations*. OpenReview.net.

[79] Michael I. Jordan. 1997. Chapter 25 - serial order: A parallel distributed processing approach. In *Proceedings of the Neural-Network Models of Cognition*. John W. Donahoe and Vivian Packard Dorsel (Eds.), Advances in Psychology, Vol. 121. North-Holland, 471–495.

[80] Dan Jurafsky and James H. Martin. 2022. Speech and language processing: An Introduction to natural language processing, computational linguistics, and speech recognition (3rd ed.). (2022). Draft. https://web.stanford.edu/~jurafsky/slp3/.

[81] Lukasz Kaiser, Samy Bengio, Aurko Roy, Ashish Vaswani, Niki Parmar, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. In *Proceedings of the 35th International Conference on Machine Learning*. Jennifer G. Dy and Andreas Krause (Eds.), PMLR, 2395–2404.

[82] Daejoong Kim, Jaedeug Choi, Kee-Eung Kim, Jungsu Lee, and Jinho Sohn. 2013. Engineering statistical dialog state trackers: A case study on DSTC. In *Proceedings of the SIGDIAL 2013 Conf., The 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. The Association for Computer Linguistics, 462–466.

[83] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd Int. Conf. on Learning Representations*, Yoshua Bengio and Yann LeCun (Eds.), arXiv:1412.6980. Retrieved from http://arxiv.org/abs/1412.6980.

[84] Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the 2nd Int. Conf. on Learning Representations*. Yoshua Bengio and Yann LeCun (Eds.).

[85] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the Advances in Neural Information Processing Systems 28: Annual Conf. on Neural Information Processing Systems 2015*. Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.), 3294–3302.

[86] Mojtaba Komeili, Kurt Shuster, and Jason Weston. 2021. Internet-augmented dialogue generation. arXiv:2107.07566. Retrieved from https://arxiv.org/abs/2107.07566.

[87] Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conf. on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations*. Eduardo Blanco and Wei Lu (Eds.), Association for Computational Linguistics, 66–71.

[88] Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2021. Dialogue state tracking with a language model using schema-driven prompting. In *Proceedings of the 2021 Conf. on Empirical Methods in Natural Language Processing*. Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.), Association for Computational Linguistics, 4937–4949.

[89] Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conf. on Empirical Methods in Natural Language Processing*. Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.), Association for Computational Linguistics, 1173–1182.

[90] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.), Association for Computational Linguistics, 7871–7880. DOI: https://doi.org/10.18653/v1/2020.acl-main.703

[91] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the NAACL HLT 2016, The 2016 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Kevin Knight, Ani Nenkova, and Owen Rambow (Eds.), The Association for Computational Linguistics, 110–119.

[92] Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and William B. Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7–12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

[93] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conf. on Empirical Methods in Natural Language Processing*. Jian Su, Xavier Carreras, and Kevin Duh (Eds.), The Association for Computational Linguistics, 1192–1202.

[94] Margaret Li, Stephen Roller, Ilia Kulikov, Sean Welleck, Y.-Lan Boureau, Kyunghyun Cho, and Jason Weston. 2020. Don't say that! Making inconsistent dialogue unlikely with unlikelihood training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.), Association for Computational Linguistics, 4715–4728.

[95] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A manually labelled multi-turn dialogue dataset. In *Proceedings of the 8th Int. Joint Conf. on Natural Language Processing (Volume 1: Long Papers)*. Asian Federation of Natural Language Processing, Taipei, Taiwan, 986–995.

[96] Jindrich Libovický and Jindrich Helcl. 2018. End-to-end non-autoregressive neural machine translation with connectionist temporal classification. In *Proceedings of the 2018 Conf. on Empirical Methods in Natural Language Processing*. Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.), Association for Computational Linguistics, 3016–3021.

[97] Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of the Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81.

[98] Zhaojiang Lin, Andrea Madotto, Jamin Shin, Peng Xu, and Pascale Fung. 2019. MoEL: Mixture of empathetic listeners. In *Proceedings of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing*. Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.), Association for Computational Linguistics, 121–132.

[99] Zhaojiang Lin, Peng Xu, Genta Indra Winata, Farhad Bin Siddique, Zihan Liu, Jamin Shin, and Pascale Fung. 2020. CAiRE: An end-to-end empathetic chatbot. In *Proceedings of the 34th AAAI Conf. on Artificial Intelligence, AAAI 2020, The 32nd Innovative Applications of Artificial Intelligence Conf., IAAI 2020, The 10th AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 13622–13623.

[100] Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conf. on Empirical Methods in Natural Language Processing*. Jian Su, Xavier Carreras, and Kevin Duh (Eds.), The Association for Computational Linguistics, 2122–2132.

[101] Qi Liu, Matt J. Kusner, and Phil Blunsom. 2020. A survey on contextual embeddings. arXiv:2003.07278. Retrieved from https://arxiv.org/abs/2003.07278.

[102] Qi Liu, Lei Yu, Laura Rimell, and Phil Blunsom. 2021. Pretraining the noisy channel model for task-oriented dialogue. *Transactions of the Association for Computational Linguistics* 9 (2021), 657–674.

[103] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692. Retrieved from http://arxiv.org/abs/1907.11692.

[104] Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*. Regina Barzilay and Min-Yen Kan (Eds.), Association for Computational Linguistics, 1116–1126.

[105] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the SIGDIAL 2015 Conf., The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. The Association for Computer Linguistics, 285–294.

[106] Andrea Madotto, Etsuko Ishii, Zhaojiang Lin, Sumanth Dathathri, and Pascale Fung. 2020. Plug-and-play conversational models. In *Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16–20 November 2020 (Findings of ACL, Vol. EMNLP 2020)*. Trevor Cohn, Yulan He, and Yang Liu (Eds.), Association for Computational Linguistics, 2422–2433.

[107] Ganeshan Malhotra, Abdul Waheed, Aseem Srivastava, Md. Shad Akhtar, and Tanmoy Chakraborty. 2022. Speaker and time-aware joint contextual learning for dialogue-act classification in counselling conversations. In *Proceedings of the WSDM'22: The 15th ACM Int. Conf. on Web Search and Data Mining, Virtual Event / Tempe*. K. Selcuk Candan, Huan Liu, Leman Akoglu, Xin Luna Dong, and Jiliang Tang (Eds.), ACM, 735–745. DOI: https://doi.org/10.1145/3488560.3498509

[108] Warren Mcculloch and Walter Pitts. 1943. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, 4 (1943), 127–147.

[109] Gary McKeown, Michel François Valstar, Roderick Cowie, and Maja Pantic. 2010. The SEMAINE corpus of emotionally coloured character interactions. In *Proceedings of the 2010 IEEE Int. Conf. on Multimedia and Expo*. IEEE Computer Society, 1079–1084.

[110] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *Proceedings of the 6th Int. Conf. on Learning Representations*. OpenReview.net.

[111] Microsoft. 2023. The New Bing. Retrieved from https://www.bing.com/new.

[112] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the 1st Int. Conf. on Learning Representations*. Yoshua Bengio and Yann LeCun (Eds.).

[113] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Advances in Neural Information Processing Systems 26: 27th Annual Conf. on Neural Information Processing Systems 2013*. Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.), 3111–3119.

[114] Nasrin Mostafazadeh, Chris Brockett, Bill Dolan, Michel Galley, Jianfeng Gao, Georgios P. Spithourakis, and Lucy Vanderwende. 2017. Image-grounded conversations: Multimodal context for natural question and response generation. In *Proceedings of the 8th Int. Joint Conf. on Natural Language Processing*. Greg Kondrak and Taro Watanabe (Eds.), Asian Federation of Natural Language Processing, 462–472.

[115] Multiple authors. 2013. Counseling and Psychotherapy Transcripts: Volume II. Retrieved from https://search.alexanderstreet.com/ctrn.

[116] Rostislav Nedelchev, Jens Lehmann, and Ricardo Usbeck. 2020. Language model transformers as evaluators for open-domain dialogues. In *Proceedings of the 28th Int. Conf. on Computational Linguistics*. Donia Scott, Núria Bel, and Chengqing Zong (Eds.), Int. Committee on Computational Linguistics, 6797–6808.

[117] Huyen T. M. Nguyen and David Morales. 2017. A Neural Chatbot with Personality. (2017). https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2761115.pdf.

[118] OpenAI. 2022. Introducing ChatGPT. Retrieved from https://openai.com/blog/chatgpt/.

[119] OpenAI. 2023. ChatGPT Plugins. Retrieved from https://openai.com/blog/chatgpt-plugins.

[120] OpenAI. 2023. GPT-4 technical report. arXiv:2303.08774. Retrieved from https://arxiv.org/abs/2303.08774.

[121] Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-Gram features. In *Proceedings of the 2018 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Marilyn A. Walker, Heng Ji, and Amanda Stent (Eds.), Association for Computational Linguistics, 528–540.

[122] Ana Paiva, Iolanda Leite, Hana Boukricha, and Ipke Wachsmuth. 2017. Empathy in virtual agents and robots: A survey. *ACM Trans. Interact. Intell. Syst.* 7, 3 (2017), 11:1–11:40. DOI: https://doi.org/10.1145/2912150

[123] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. ACL, 311–318.

[124] Seongmin Park and Jihwa Lee. 2021. Finetuning pretrained transformers into variational autoencoders. In *Proceedings of the 2nd Workshop on Insights from Negative Results in NLP*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 29–35.

[125] Razvan Pascanu, Tomás Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th Int. Conf. on Machine Learning*. JMLR.org, 1310–1318.

[126] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conf. on Empirical Methods in Natural Language Processing*. Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.), ACL, 1532–1543.

[127] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Marilyn A. Walker, Heng Ji, and Amanda Stent (Eds.), Association for Computational Linguistics, 2227–2237. DOI: https://doi.org/10.18653/v1/n18-1202

[128] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick S. H. Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: A benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.), Association for Computational Linguistics, 2523–2544. DOI: https://doi.org/10.18653/v1/2021.naacl-main.200

[129] Rosalind W. Picard. 1997. *Affective Computing*. MIT press.

[130] Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2019. MELD: A multimodal multi-party dataset for emotion recognition in conversations. In *Proceedings of the 57th Conf. of the Association for Computational Linguistics*. Anna Korhonen, David R. Traum, and Lluís Màrquez (Eds.), Association for Computational Linguistics, 527–536.

[131] David M. W. Powers. 1998. The total turing test and the loebner prize. In *Proceedings of the Joint Conf. on New Methods in Language Processing and Computational Natural Language Learning*. David M. W. Powers (Ed.), ACL, 279–280.

[132] Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conf. of the European Chapter of the Association for Computational Linguistics*. Mirella Lapata, Phil Blunsom, and Alexander Koller (Eds.), Association for Computational Linguistics, 157–163.

[133] Markus N. Rabe and Charles Staats. 2021. Self-attention does not need $O(n^2)$ memory. arXiv:2112.05682. Retrieved from https://arxiv.org/abs/2112.05682.

[134] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Blog* 1, 11 (2018), 12.

[135] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* 1, 8 (2019), 9.

[136] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling language models: Methods, analysis & insights from training gopher. arXiv:2112.11446. Retrieved from https://arxiv.org/abs/2112.11446.

[137] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research* 21 (2020), 140:1–140:67. https://jmlr.org/papers/v21/20-074.html.

[138] Ashwin Ram, Rohit Prasad, Chandra Khatri, Anu Venkatesh, Raefer Gabriel, Qing Liu, Jeff Nunn, Behnam Hedayatnia, Ming Cheng, Ashish Nagar, Eric King, Kate Bland, Amanda Wartick, Yi Pan, Han Song, Sk Jayadevan, Gene Hwang, and Art Pettigrue. 2018. Conversational AI: The science behind the alexa prize. arXiv:1801.03604. Retrieved from https://arxiv.org/abs/1801.03604.

[139] Justus J. Randolph. 2005. Free-marginal multirater kappa (multirater κfree): An alternative to fleiss fixed-marginal multirater kappa. In *Proceedings of the Joensuu Learning and Instruction Symposium*. 1–20.

[140] Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y.-Lan Boureau. 2019. Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 5370–5381.

[141] Scott E. Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. 2022. A generalist agent. arXiv:2205.12478. Retrieved from https://arxiv.org/abs/2205.12478.

[142] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing*. Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.), Association for Computational Linguistics, 3980–3990.

[143] D. E. Remelhart, G. E. Hinton, and J. Williams. 1986. Learning Internal representations by error backpropagation. In *Proceedings of the Parallel Distributed Process*. The MIT press.

[144] Alan Ritter, Colin Cherry, and William B. Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the 2011 Conf. on Empirical Methods in Natural Language Processing*. ACL, 583–593.

[145] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y.-Lan Boureau, and Jason Weston. 2021. Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conf. of the European Chapter of the Association for Computational Linguistics: Main Volume*. Paola Merlo, Jörg Tiedemann, and Reut Tsarfaty (Eds.), Association for Computational Linguistics, 300–325.

[146] Frank Rosenblatt. 1961. *Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms*. Technical Report. Cornell Aeronautical Lab Inc Buffalo NY.

[147] Waddah Saeed and Christian Omlin. 2023. Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems* 263 (2023), 110273. https://www.sciencedirect.com/science/article/pii/S0950705123000230.

[148] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M. Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *Proceedings of the 10th Int. Conf. on Learning Representations*. OpenReview.net. Retrieved from https://openreview.net/forum?id=9Vrb9D0WI4.

[149] Chinnadhurai Sankar and Sujith Ravi. 2019. Deep reinforcement learning for modeling chit-chat dialog with discrete attributes. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*. Satoshi Nakamura, Milica Gasic, Ingrid Zuckerman, Gabriel Skantze, Mikio Nakano, Alexandros Papangelis, Stefan Ultes, and Koichiro Yoshino (Eds.), Association for Computational Linguistics, 1–10.

[150] Bishal Santra, Potnuru Anusha, and Pawan Goyal. 2021. Hierarchical transformer for task oriented dialog systems. In *Proceedings of the 2021 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.), Association for Computational Linguistics, 5649–5658.

[151] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilic, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani et al. 2022. BLOOM: A 176B-parameter open-access multilingual language model. arXiv:2211.05100. Retrieved from https://arxiv.org/abs/2211.05100.

[152] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. arXiv:1707.06347. Retrieved from https://arxiv.org/abs/1707.06347.

[153] Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* 45, 11 (1997), 2673–2681.

[154] João Sedoc, Daphne Ippolito, Arun Kirubarajan, Jai Thirani, Lyle Ungar, and Chris Callison-Burch. 2019. ChatEval: A tool for chatbot evaluation. In *Proceedings of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Waleed Ammar, Annie Louis, and Nasrin Mostafazadeh (Eds.), Association for Computational Linguistics, 60–65. DOI: https://doi.org/10.18653/v1/n19-4011

[155] Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019. What makes a good conversation? How controllable attributes affect human judgments. In *Proceedings of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Jill Burstein, Christy Doran, and Thamar Solorio (Eds.), Association for Computational Linguistics, 1702–1723.

[156] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. The Association for Computer Linguistics.

[157] Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. 2018. A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue Discourse* 9, 1 (2018), 1–49.

[158] Iulian Vlad Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, Sai Mudumba, Alexandre de Brébisson, Jose Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen, Joelle Pineau, and Yoshua Bengio. 2017. A deep reinforcement learning chatbot. arXiv:1709.02349. Retrieved from https://arxiv.org/abs/1709.02349.

[159] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conf. on Artificial Intelligence*. Dale Schuurmans and Michael P. Wellman (Eds.), AAAI Press, 3776–3784.

[160] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the 31st AAAI Conf. on Artificial Intelligence*. Satinder P. Singh and Shaul Markovitch (Eds.), AAAI Press, 3295–3301.

[161] Lifeng Shang, Tetsuya Sakai, Hang Li, Ryuichiro Higashinaka, Yusuke Miyao, Yuki Arase, and Masako Nomoto. 2017. Overview of the NTCIR-13 short text conversation task.. In *Proceedings of the NTCIR (CEUR Workshop Proceedings, Vol. 2008)*, Nicola Ferro and Ian Soboroff (Eds.), CEUR-WS.org.

[162] Lifeng Shang, Tetsuya Sakai, Zhengdong Lu, Hang Li, Ryuichiro Higashinaka, and Yusuke Miyao. 2016. Overview of the NTCIR-12 short text conversation task. In *Proceedings of the 12th NTCIR Conf. on Evaluation of Information Access Technologies, National Center of Sciences*. Noriko Kando, Tetsuya Sakai, and Mark Sanderson (Eds.), National Institute of Informatics (NII).

[163] Ashish Sharma, Inna W. Lin, Adam S. Miner, David C. Atkins, and Tim Althoff. 2021. Towards facilitating empathic conversations in online mental health support: A reinforcement learning approach. In *Proceedings of the WWW '21: The Web Conf. 2021*. Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.), ACM / IW3C2, 194–205. DOI: https://doi.org/10.1145/3442381.3450097

[164] Ashish Sharma, Adam S. Miner, David C. Atkins, and Tim Althoff. 2020. A computational approach to understanding empathy expressed in text-based mental health support. In *Proceedings of the 2020 Conf. on Empirical Methods in Natural Language Processing*. Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.), Association for Computational Linguistics, 5263–5276. DOI: https://doi.org/10.18653/v1/2020.emnlp-main.425

[165] Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th Int. Conf. on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*. Jennifer G. Dy and Andreas Krause (Eds.), PMLR, 4603–4611. Retrieved from http://proceedings.mlr.press/v80/shazeer18a.html.

[166] Jamin Shin, Peng Xu, Andrea Madotto, and Pascale Fung. 2020. Generating empathetic responses by looking ahead the user's sentiment. In *Proceedings of the 2020 IEEE Int. Conf. on Acoustics, Speech and Signal Processing*. IEEE, 7989–7993.

[167] Kurt Shuster, Samuel Humeau, Antoine Bordes, and Jason Weston. 2018. Engaging image chat: Modeling personality in grounded dialogue. arXiv:1811.00945. Retrieved from https://arxiv.org/abs/1811.00945.

[168] Kurt Shuster, Da Ju, Stephen Roller, Emily Dinan, Y.-Lan Boureau, and Jason Weston. 2020. The dialogue dodecathlon: Open-domain knowledge and image grounded conversational agents. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.), Association for Computational Linguistics, 2453–2470.

[169] Kurt Shuster, Jing Xu, Mojtaba Komeili, Da Ju, Eric Michael Smith, Stephen Roller, Megan Ung, Moya Chen, Kushal Arora, Joshua Lane, Morteza Behrooz, William Ngan, Spencer Poff, Naman Goyal, Arthur Szlam, Y.-Lan Boureau, Melanie Kambadur, and Jason Weston. 2022. BlenderBot 3: A deployed conversational agent that continually learns to responsibly engage. arXiv:2208.03188. Retrieved from https://arxiv.org/abs/2208.03188.

[170] Hava T. Siegelmann and Eduardo D. Sontag. 1995. On the computational power of neural nets. *Journal of Computer and System Sciences* 50, 1 (1995), 132–150.

[171] Eric Michael Smith, Diana Gonzalez-Rico, Emily Dinan, and Y.-Lan Boureau. 2020. Controlling style in generated dialogue. arXiv:2009.10855. Retrieved from https://arxiv.org/abs/2009.10855.

[172] Eric Michael Smith, Orion Hsu, Rebecca Qian, Stephen Roller, Y.-Lan Boureau, and Jason Weston. 2022. Human evaluation of conversations is an open problem: Comparing the sensitivity of various methods for evaluating dialogue agents. In *Proceedings of the 4th Workshop on NLP for Conversational AI*. Bing Liu, Alexandros Papangelis, Stefan Ultes, Abhinav Rastogi, Yun-Nung Chen, Georgios Spithourakis, Elnaz Nouri, and Weiyan Shi (Eds.), Association for Computational Linguistics, 77–97. DOI: https://doi.org/10.18653/v1/2022.nlp4convai-1.8

[173] Eric Michael Smith, Mary Williamson, Kurt Shuster, Jason Weston, and Y.-Lan Boureau. 2020. Can you put it all together: Evaluating conversational agents' ability to blend skills. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*. Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.), Association for Computational Linguistics, 2021–2030. DOI: https://doi.org/10.18653/v1/2020.acl-main.183

[174] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Proceedings of the Advances in Neural Information Processing Systems 28: Annual Conf. on Neural Information Processing Systems 2015*. Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (Eds.), 3483–3491.

[175] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the NAACL HLT 2015, The 2015 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Rada Mihalcea, Joyce Yue Chai, and Anoop Sarkar (Eds.), The Association for Computational Linguistics, 196–205.

[176] Aseem Srivastava, Tharun Suresh, Sarah Peregrine Lord, Md. Shad Akhtar, and Tanmoy Chakraborty. 2022. Counseling summarization using mental health knowledge guided utterance filtering. In *Proceedings of the KDD '22: The 28th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining*. Aidong Zhang and Huzefa Rangwala (Eds.), ACM, 3920–3930. DOI: https://doi.org/10.1145/3534678.3539187

[177] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[178] Yixuan Su and Nigel Collier. 2022. Contrastive search is what you need for neural text generation. arXiv:2210.14140. Retrieved from https://arxiv.org/abs/2210.14140.

[179] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Advances in Neural Information Processing Systems 27: Annual Conf. on Neural Information Processing Systems 2014*. Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (Eds.), 3104–3112.

[180] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA.

[181] Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. 2018. RUBER: An unsupervised method for automatic evaluation of open-domain dialog systems. In *Proceedings of the 32nd AAAI Conf. on Artificial Intelligence, the 30th innovative Applications of Artificial Intelligence, and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence*. Sheila A. McIlraith and Kilian Q. Weinberger (Eds.), AAAI Press, 722–729.

[182] Henry S. Thompson, Anne H. Anderson, Ellen Gurman Bard, Gwyneth Doherty-Sneddon, Alison Newlands, and Catherine Sotillo. 1993. THE HCRC MAP TASK CORPUS: Natural dialogue for speech recognition. In *Proceedings of the Human Language Technology: Proceedings of a Workshop Held at Plainsboro*. Morgan Kaufmann.

[183] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Kathleen S. Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed H. Chi, and Quoc Le. 2022. LaMDA: Language models for dialog applications. arXiv:2201.08239. Retrieved from https://arxiv.org/abs/2201.08239.

[184] Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. 2019. Learning to speak and act in a fantasy text adventure game. In *Proceedings of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing*. Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.), Association for Computational Linguistics, 673–683.

[185] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conf. on Neural Information Processing Systems 2017*. Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.), 6306–6315.

[186] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conf. on Neural Information Processing Systems 2017*. Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.), 5998–6008.

[187] Anu Venkatesh, Chandra Khatri, Ashwin Ram, Fenfei Guo, Raefer Gabriel, Ashish Nagar, Rohit Prasad, Ming Cheng, Behnam Hedayatnia, Angeliki Metallinou, Rahul Goel, Shaohua Yang, and Anirudh Raju. 2018. On evaluating and comparing conversational agents. arXiv:1801.03625. Retrieved from https://arxiv.org/abs/1801.03625.

[188] Eija Ventola. 1979. The structure of casual conversation in English. *Journal of Pragmatics* 3, 3 (1979), 267–298.

[189] Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. arXiv:1506.05869. Retrieved from https://arxiv.org/abs/1506.05869.

[190] Richard Wallace. 2003. The Elements of AIML Style. https://web.archive.org/web/20060510064356/http://www.alicebot.org/style.pdf.

[191] Richard S. Wallace. 2009. *The Anatomy of A.L.I.C.E.* Springer Netherlands, Dordrecht, 181–210.

[192] Alex Wang and Kyunghyun Cho. 2019. BERT has a mouth, and it must speak: BERT as a Markov random field language model. arXiv:1902.04094. Retrieved from https://arxiv.org/abs/1902.04094.

[193] Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. 2022. What language model architecture and pretraining objective works best for zero-shot generalization?. In *Proceedings of the Int. Conf. on Machine Learning*. Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.), PMLR, 22964–22984. Retrieved from https://proceedings.mlr.press/v162/wang22u.html.

[194] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *Proceedings of the 10th Int. Conf. on Learning Representations*. OpenReview.net. Retrieved from https://openreview.net/forum?id=gEZrGCozdqR.

[195] Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, Courtney Biles, Sasha Brown, Zac Kenton, Will Hawkins, Tom Stepleton, Abeba Birhane, Lisa Anne Hendricks, Laura Rimell, William S. Isaac, Julia Haas, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2022. Taxonomy of risks posed by language models. In *Proceedings of the FAccT '22: 2022 ACM Conf. on Fairness, Accountability, and Transparency*. ACM, 214–229. DOI: https://doi.org/10.1145/3531146.3533088

[196] Joseph Weizenbaum. 1983. ELIZA - A computer program for the study of natural language communication between man and machine (reprint). *Communications of the ACM* 26, 1 (1983), 23–28.

[197] Anuradha Welivita, Yubo Xie, and Pearl Pu. 2021. A large-scale dataset for empathetic response generation. In *Proceedings of the 2021 Conf. on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 1251–1264.

[198] Jason Weston, Emily Dinan, and Alexander H. Miller. 2018. Retrieve and refine: Improved sequence generation models for dialogue. In *Proceedings of the 2nd Int. Workshop on Search-Oriented Conversational AI*. Aleksandr Chuklin, Jeff Dalton, Julia Kiseleva, Alexey Borisov, and Mikhail S. Burtsev (Eds.), Association for Computational Linguistics, 87–92.

[199] Bernard Widrow. 1960. An Adaptive'adaline'neuron using chemical'memistors', 1553–1552. https://isl.stanford.edu/~widrow/papers/t1960anadaptive.pdf.

[200] Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* 8, 3 (1992), 229–256.

[201] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conf. on Empirical Methods in Natural Language Processing: System Demonstrations*. Qun Liu and David Schlangen (Eds.), Association for Computational Linguistics, 38–45.

[202] Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. TransferTransfo: A transfer learning approach for neural network based conversational agents. arXiv:1901.08149. Retrieved from https://arxiv.org/abs/1901.08149.

[203] Steve Worswick. 2018. Mitsuku wins Loebner Prize 2018! https://medium.com/pandorabots-blog/mitsuku-wins-loebner-prize-2018-3e8d98c5f2a7.

[204] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144. Retrieved from https://arxiv.org/abs/1609.08144.

[205] Jing Xu, Da Ju, Margaret Li, Y-Lan Boureau, Jason Weston, and Emily Dinan. 2020. Recipes for safety in open-domain chatbots. arXiv:2010.07079. Retrieved from https://arxiv.org/abs/2010.07079.

[206] Jing Xu, Arthur Szlam, and Jason Weston. 2021. Beyond goldfish memory: Long-term open-domain conversation. arXiv:2107.07567. Retrieved from https://arxiv.org/abs/2107.07567.

[207] Jing Xu, Megan Ung, Mojtaba Komeili, Kushal Arora, Y.-Lan Boureau, and Jason Weston. 2022. Learning new skills after deployment: Improving open-domain internet-driven dialogue with human feedback. arXiv:2208.03270. Retrieved from https://arxiv.org/abs/2208.03270.

[208] Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Trans. Assoc. Comput. Linguistics* 10 (2022), 291–306. https://aclanthology.org/2022.tacl-1.17/.

[209] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.), Association for Computational Linguistics, 483–498. DOI: https://doi.org/10.18653/v1/2021.naacl-main.41

[210] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *Proceedings of the Advances in Neural Information Processing Systems 27: Annual Conf. on Neural Information Processing Systems 2014*. Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (Eds.), 3320–3328.

[211] Rohola Zandie and Mohammad H. Mahoor. 2020. EmpTransfo: A multi-head transformer architecture for creating empathetic dialog systems. In *Proceedings of the 33rd Int. Florida Artificial Intelligence Research Society Conf.* Roman Barták and Eric Bell (Eds.), AAAI Press, 276–281.

[212] Yan Zeng and Jian-Yun Nie. 2021. A simple and efficient multi-task learning approach for conditioned dialogue generation. In *Proceedings of the 2021 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (Eds.), Association for Computational Linguistics, 4927–4939. DOI: https://doi.org/10.18653/v1/2021.naacl-main.392

[213] Zhaohao Zeng, Sosuke Kato, and Tetsuya Sakai. 2019. Overview of the NTCIR-14 short text conversation task: Dialogue quality and nugget detection subtasks. In *Proceedings of the 14th NTCIR Conf. on Evaluation of Information Access Technologies*. Yiqun Kato, Makoto P. Liu, Noriko Kando, and Charles L. A. Clarke (Eds.), Springer.

[214] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too?. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Melbourne, Australia, 2204–2213.

[215] Yaoqin Zhang and Minlie Huang. 2019. Overview of the NTCIR-14 short text generation subtask: Emotion generation challenge. In *Proceedings of the 14th NTCIR Conf.* Yiqun Kato, Makoto P. Liu, Noriko Kando, and Charles L. A. Clarke (Eds.), Springer.

[216] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DIALOGPT: Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020*. Asli Celikyilmaz and Tsung-Hsien Wen (Eds.), Association for Computational Linguistics, 270–278.

[217] Yuchi Zhang, Yongliang Wang, Liping Zhang, Zhiqiang Zhang, and Kun Gai. 2019. Improve diverse text generation by self labeling conditional variational auto encoder. In *Proceedings of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing*. IEEE, 2767–2771.

[218] Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Regina Barzilay and Min-Yen Kan (Eds.), Association for Computational Linguistics, 654–664.

[219] Vitalii Zhelezniak, Aleksandar Savkov, April Shen, Francesco Moramarco, Jack Flann, and Nils Y. Hammerla. 2019. Don't settle for average, go for the max: Fuzzy sets and max-pooled word vectors. In *Proceedings of the 7th Int. Conf. on Learning Representations*. OpenReview.net.

[220] Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. 2020. The design and implementation of XiaoIce, an empathetic social chatbot. *Comput. Linguistics* 46, 1 (2020), 53–93.

[221] Qile Zhu, Wei Bi, Xiaojiang Liu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. 2020. A batch normalized inference network keeps the KL vanishing away. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.), Association for Computational Linguistics, 2636–2649.

[222] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 IEEE Int. Conf. on Computer Vision*. IEEE Computer Society, 19–27.

Received 16 June 2022; revised 6 May 2023; accepted 31 May 2023