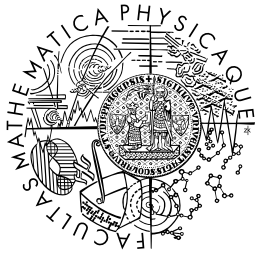


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. František Trebuňa

Persona-Aware Chatbot Response Generation

Institute of Formal and Applied Linguistics

Supervisor of the master thesis:	Mgr. et Mgr. Ondřej Dušek, Ph.D.
Study programme:	Computer Science - Artificial Intelligence
Study branch:	Computer Science - Artificial Intelligence

Prague 2024

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I would like to express my gratitude to my family and my girlfriend for their support. I am also thankful to my supervisor, Mgr. et Mgr. Ondřej Dušek, Ph.D., for his invaluable guidance and feedback.

Title: Persona-Aware Chatbot Response Generation

Author: Bc. František Trebuňa

Institute: Institute of Formal and Applied Linguistics

Supervisor: Mgr. et Mgr. Ondřej Dušek, Ph.D., Institute of Formal and Applied Linguistics

Abstract: This thesis investigates non-task-oriented open-domain dialogue modeling, i.e., generating responses in a basic social conversation, using the ConvAI2 dataset, focusing on using neural language models and maintaining consistent chatbot's personality and overall coherence. We work with finetuning models from the GPT-2 family; we improve over a basic finetuned generation setup with a two-stage approach and an additional model learned to rank candidates by the first model. We further improve model training using direct preference optimization. Our modifications achieve state-of-the-art performance in the F1 score on the ConvAI2 dataset. We then engineer a prompt for GPT-3.5 to use this large language model for the task. Human evaluation experiment reveals that, despite lower F1 scores, the GPT-3.5 model surpasses all others in performance.

Keywords: chatbots, dialogue, response generation, natural language processing, neural language models

Contents

Introduction

In recent years, the field of dialogue systems has seen significant advancements, particularly in the area of end-to-end dialogue modeling based on pretrained neural language models [Wolf et al., 2019, Zhang et al., 2020, Adiwardana et al., 2020], including the latest large language models [Wei et al., 2022, Ouyang et al., 2022, OpenAI, 2023]. This thesis focuses on open-domain non-task-oriented dialogue, i.e., on a general social conversation between a dialogue model and a human that does not follow any particular goal except entertainment (Table 1).

Who	Turn
A	hello how are you this gorgeous day ?
B	i'm great ! how are you ?
A	fairly good . just trying to decide what i'm going to cook tonight
B	i see . what do you like to eat ?
A	i am a gourmet cook so i make everything

Table 1: An excerpt from a dialogue from the ConvAI2 dataset [Dinan et al., 2020]. See Chapter 2 for more details on the data.

While neural end-to-end dialogue models can produce very fluent outputs and are able to produce previously unseen sentences, they often generate incoherent responses and do not maintain a consistent personality [Zhang et al., 2018, Li et al., 2016c]. This thesis aims to overcome these problems by finetuning small language models and engineering a prompt for large language models to perform well on the popular ConvAI2 dataset [Dinan et al., 2020], a common benchmark for non-task-oriented dialogue.

We aim to answer the following research questions:

1. Can the performance of small language models finetuned on the ConvAI2 dataset be improved through a two-stage approach, where the finetuned model generates several response candidates in the first stage, and a ranking model selects the final response in the second stage?

2. Is it possible to improve the finetuned models by more sophisticated training schemes, such as applying direct preference optimization [Rafailov et al., 2023] on a preference dataset?

3. How do prompted large language models compare to finetuned small language models on the ConvAI2 dataset?

Chapter 1 introduces the reader to dialogue modeling with neural networks. We follow by presenting the ConvAI2 dataset (Chapter 2) and evaluation methods that are used to evaluate the performance of open-domain dialogue models (Chapter 3). We show other approaches that were used for dialogue modeling on the ConvAI2 dataset in Chapter 4, and finally, we discuss our own experiments and provide a short summarizing conclusion in Chapter 5.

1. Theoretical Background

This chapter introduces the reader to several concepts which are necessary to understand to follow our experiments. We begin with a general introduction to neural networks (Section 1.1). We follow by discussing neural network architectures that are commonly used for natural language generation (Section 1.2) and introducing the reader to the dialogue modeling task (Section 1.3).

For several years, the pretraining and finetuning discussed in Section 1.4, was the approach that was the most effective for natural language generation. However, recently a prompting approach, where a pretrained large language model is instructed to solve a task by natural language, emerged as a viable alternative to pretraining and finetuning. We discuss prompting in Section 1.5. We conclude by introducing the reader to the problematic of training of ranking models (Section 1.6) that we use in our experiments in Chapter 5.

1.1 Neural Networks

We do not intend to explore the theoretical properties of neural networks in this text. Hence, this section only presents a few basic concepts from neural networks that are needed to understand concepts related to training of neural neural networks which we conduct in Chapter 5. The text in this section is heavily inspired by [Goodfellow et al., 2016].

Definition of a Neural Network A neural network is a mapping $\mathbf{y} = f(\mathbf{x}; \theta)$, where x is an input of the neural network (e.g., a sequence of words), and y is the output (e.g., probability that the sequence is of positive sentiment). The parameterized function f represents the architecture of the network and has to be specified manually.[1] The choice of f affects the class of functions that can be approximated by altering the parameters θ. E.g., by using $f(\mathbf{x}, \theta) = \theta^\top \mathbf{x}$ only linear functions can be approximated.

Supervision In the most common case, the true function that we want to learn is either not known or is intractable and we only have access to a dataset of examples. Based on the examples in the dataset, we distinguish two types of training methods.

- Unsupervised learning algorithms experience a dataset $\mathcal{D} = \{\mathbf{x_i}\}_{i=1}^N$. The goal of an unsupervised method is to learn properties of this dataset, e.g., the probability distribution that generated the dataset.
- Supervised learning algorithms make use of datasets $\mathcal{D} = \{(\mathbf{x_i}, \mathbf{y_i})\}_{i=1}^N$, where $\mathbf{x_i}$ are inputs and $\mathbf{y_i}$ are target outputs. A supervised method learns to predict the target from a feature.

In the following text we only discuss only supervised methods, since in this thesis we do not make use of any unsupervised method.

[1] While approaches that automatically optimize the network architecture exist, this is outside the scope of this thesis.

Multi-Layer Perceptron The multi-layer perceptron is the simplest neural network architecture. The computation of each layer is composed of an affine transformation controlled by learned parameters, followed by a nonlinear *activation function*. The output of layer n serves as an input to layer $n+1$. Equation 1.1 shows a computation of a two layer perceptron. Since the information about the input can only go from layer n to layer $n+1$, the multi-layer perceptron is sometimes called *feed-forward network*.

$$f(\mathbf{x};\theta) = a_2(b_{\theta,2} + W_{\theta,2} \cdot a_1(b_{\theta,1} + W_{\theta,1} \cdot x)) \tag{1.1}$$

The activation function at the outer-most layer is chosen to complete the task that the network must perform. Here, we list the most common output activation functions:

- linear: commonly used to produce the mean of a conditional gaussian distribution
- sigmoid: estimation of a Bernoulli distribution

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{1.2}$$

- softmax: estimation of a Multinoulli distribution (an extension of a Bernoulli distribution to multiple classes)

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{1.3}$$

There are other more complex neural network architectures such as convolutional neural networks [LeCun et al., 1989], recurrent neural networks [Rumelhart et al., 1986] (discussed in Section 1.2.3), or the transformer architecture [Vaswani et al., 2017] (presented in Section 1.2.7).

Maximum Likelihood Estimation In order to find parameters θ which minimize the dissimilarity between the estimated distribution $p_{\text{model}}(\mathbf{y}|\mathbf{x};\theta)$ and the distribution $p_{\text{data}}(\mathbf{y}|\mathbf{x})$ we use the *maximum likelihood principle*. The maximum likelihood principle refers to an optimization problem where we search for parameters θ_{ML} such that the estimated conditional probability of the dataset $\mathcal{D}$ given θ_{ML} is maximized (Equation 1.4).

$$\theta_{\text{ML}} = \arg\max_{\theta} p_{\text{model}}(\mathcal{D}_y|\mathcal{D}_x;\theta) \tag{1.4}$$

Assuming that examples in the dataset are identically distributed (i.e., there is a single a distribution p_{data} such that $(\mathbf{x_i}, \mathbf{y_i}) \sim p_{\text{data}} \forall i$) and independent from each other (i.e., $p_{\text{data}}(\mathcal{D}) = \prod_{i=1}^{N} p_{\text{data}}(\mathbf{x_i}, \mathbf{y_i})$), we can describe the data-generating process (i.e., probability distribution over datasets) with a probability distribution over single example $p_{\text{data}}(\mathbf{y_i}, \mathbf{x_i})$.

Therefore, we can reformulate the estimated probability of the dataset $p_{\text{model}}(\mathcal{D}_y|\mathcal{D}_x;\theta)$ as a product of estimated conditional probabilities of samples (Equation 1.5).

$$\theta_{\text{ML}} = \arg\max_{\theta} \prod_{i=1}^{N} p_{\text{model}}(\mathbf{y_i}|\mathbf{x_i};\theta) \tag{1.5}$$

Taking a logarithm of likelihood does not change the optimization problem (since logarithm is a monotonic function) and summation has better properties with respect to numerical underflow. Moreover multiplication by -1 allows the use of a minimization formulation (Equation 1.6).

$$\theta_{\text{ML}} = \arg\min_{\theta} -\sum_{i=1}^{N} \log p_{\text{model}}(\mathbf{y_i}|\mathbf{x_i};\theta) \tag{1.6}$$

Since minimization does not change with scaling by a positive number, a division by N allows us to formulate the problem as minimization of negative expectation with respect to the data-generating distribution. The function $\mathcal{L}$ in Equation 1.7 is commonly named *negative log likelihood*, or *cross-entropy* H between distributions p_{data} and p_{model}.

$$\begin{aligned} \mathcal{L}(\theta) &= -\mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim p_{\text{data}}} \log p_{\text{model}}(\mathbf{y_i}|\mathbf{x_i};\theta) \\ \theta_{\text{ML}} &= \arg\min_{\theta} \mathcal{L}(\theta) \end{aligned} \tag{1.7}$$

Gradient Descent The optimization problem in Equation 1.7 is intractable for real-world data-generating distributions, i.e., it is not possible to calculate its closed-form solution. To solve, or approximate a solution of Equation 1.7 a numerical computation algorithms are used.

Gradient-descent-based optimization algorithms make use of an observation that we can solve the minimization problem in an infinitesimally small local neighborhood of θ. In this neighborhood, the minimum is attained at $\theta^* = \theta - \lim_{\alpha\to 0} \alpha\nabla_\theta f(\theta)$, i.e., by taking an infinitesimally small step in the direction opposite to the direction of gradient.

During one iteration of gradient descent, the parameters of the network are updated by taking a small step in the direction opposite to gradient. The size of the step is controlled by a hyper-parameter ϵ, the *learning rate*.

$$\theta \leftarrow \theta - \epsilon\nabla_\theta f(\theta) \tag{1.8}$$

There exists a wast array of different gradient-descent-based optimization algorithms, such as Stochastic Gradient Descent (SGD) [Bottou, 1999], SGD with momentum [Polyak, 1964], SGD with Nesterov momentum [Sutskever et al., 2013], AdaGrad [Duchi et al., 2011], or Adam [Kingma and Ba, 2015] that uses the estimates of first and second moments of gradients (mean and variance of gradients) to optimize neural network training. In this thesis we use only the Adam optimizer as it is a widely accepted standard choice.

1.2 Deep Learning in Natural Language Generation

Deep neural networks have achieved success in various tasks, with image classification being one of the first notable applications. Architectures such as AlexNet [Krizhevsky et al., 2012], VGG [Simonyan and Zisserman, 2015] and ResNet [He et al., 2016] have achieved state-of-the-art performance in image recognition and classification tasks.

In order to use deep neural networks for natural language generation (NLG), several key concepts had to be explored. In this section, we discuss some of the most important ideas that lead to the success of neural networks, including large language models, in this task. First, we explore how a text is transformed into continuous vectors processable by a neural network (Sections 1.2.1, 1.2.2. Then, we discuss the basics of language modeling with neural networks (Sections 1.2.3-1.2.5). We conclude with an overview of architectures that are used for language modeling (Sections 1.2.6, 1.2.7).

1.2.1 Tokenization

Due to combinatorial explosion, it is infeasible to process texts as atomic units. Tokenization refers to splitting a text into smaller units – tokens. Two natural tokenization schemes, i.e., tokenization to words and to individual characters have numerous problems.

Word-level tokenization is ineffective for compounds (e.g., in the case of comparatives, separate vocabulary entries are needed for adjective, comparative and superlative form, while a simpler composition of adjective and "er" / "est" is more effective). Since a neural network learns a separate representation of each unique token (see Section 1.2.2), its vocabulary size is typically limited to 30-60,000 entries in practice [Radford and Narasimhan, 2018, Radford et al., 2019]. Hence, special treatment is needed for rare words (e.g., substitution of new words unseen at training time by a special <UNK> token). Lastly, word-level language models cannot generate new, unseen words.

On the other hand, while a character-level model can process rare words with a smaller vocabulary, sequences of character-level tokens are much longer than sequences of word-level tokens and hence more computation and ability to model longer dependencies in text is needed compared to word-level models.

Sub-word tokenization [Sennrich et al., 2016, Kudo and Richardson, 2018] overcomes all these issues by splitting the text into subwords. In Byte Pair Encoding (BPE) tokenization [Sennrich et al., 2016] the process of splitting text is learned. Initially, the vocabulary contains solely of individual characters. During training, the most frequently adjacent tokens in the text are iteratively merged into single tokens, expanding the vocabulary. Sennrich et al. [2016] note that with the BPE tokenization scheme, any word can be expressed (similarly to character-level tokenization) while the number of tokens is reduced five-fold compared to character-level representation.

1.2.2 Embeddings

The first step in processing a sequence of tokens with a neural network is transforming each token into an embedding, i.e., a vector or a point in a high-dimensional space.

Neural language models discussed in this thesis use an embedding layer, a mapping of token indices to embeddings [Bengio et al., 2003]. While it is possible to initialize the embedding layer with pretrained embeddings [Dai and Le, 2015] (examples of pretrained embeddings include Word2vec [Mikolov et al., 2013], GloVe [Pennington et al., 2014], or FastText [Bojanowski et al., 2017]), a common practice is to initialize them to random vectors and pretrain them jointly with the model [Radford and Narasimhan, 2018, Radford et al., 2019]. In comparison to the vocabulary size of tokenizers, embeddings are low-dimensional (a typical embedding has between 768 and 12288 dimensions [Devlin et al., 2019, Brown et al., 2020]). Embeddings have some interesting properties, e.g., the embeddings of words of words commonly used in the same context tend to be nearby in the embedding space [Li et al., 2016a].

1.2.3 Neural Language Models

Sutskever et al. [2014] note that "Deep neural networks can only be applied to problems whose inputs and targets can be sensibly encoded with vectors of fixed dimensionality." Models such as multi-layer perceptron (discussed in Section 1.1) cannot be used for estimating probability of sequences of unknown length. We first show a general formalization of language modeling, then its application in a recurrent neural network paradigm.

Formalization: A language model represents a probability distribution over sequences of tokens $x_1, ..., x_N$ [Jurafsky and Martin, 2024, Chap.-3].

$$p(x_1, ..., x_N) = p(x_{1:N}) \tag{1.9}$$

Using the chain rule of probability, we can decompose the distribution of the sequence of tokens to the product of conditional next token distributions $p(x_i|x_{1:i-1})$, where $x_{1:0} = \emptyset$

$$\begin{aligned} p(x_{1:N}) &= p(x_N|x_{1:N-1})p(x_{N-1}|x_{1:N-2})...p(x_1) \\ &= \prod_{i=1}^{N} p(x_i|x_{1:i-1}) \end{aligned} \tag{1.10}$$

Recurrent Neural Networks A recurrent neural network [Rumelhart et al., 1986] processes input tokens (by their embeddings) one-by-one. At each step, two representations of already processed tokens are produced: (1) the *hidden state* h_t refers to a state that is kept and updated during the whole processing of a sequence and stores a latent representation of the sequence which is passed on as an input to the next computation step. The hidden state h_0 is initialized to a zero vector. (2) the *output representation* $\hat{y}$ (and the output activation function a_y) is task specific (e.g., for language models $a_y = \text{softmax}$).

$$
\begin{aligned}
h_{t+1} &= a_h(W_{hh}h_{t-1} + W_{hx}x_t + b_h) \\
\hat{y}_t &= a_y(W_{yh}h_t + b_y)
\end{aligned}
\tag{1.11}
$$

A standard RNN, composed of a single sigmoid/tanh layer, was superseded by more robust cells such as LSTM [Hochreiter and Schmidhuber, 1997], or GRU [Cho et al., 2014]. The whole RNN paradigm for language modelling has been later replaced by a non-recurrent transformer architecture [Vaswani et al., 2017] (discussed in Section 1.2.7).

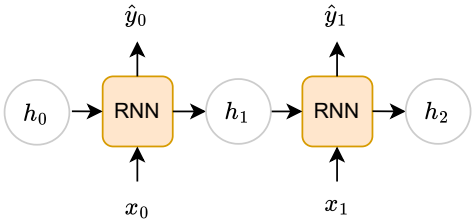


Figure 1.1: Computation of an RNN cell. Weights are shared between two RNN cells in the image.

Training Neural language models are trained to estimate the conditional next token distribution $p(x_i|x_{1:i-1})$. During training, the cross-entropy between the estimated distribution $p_{LM}(x_i|x_{1:i-1})$ and the true, one-hot, token distribution $p^*(x_i|x_{1:i-1})$ (discussed in Section 1.1) is used as loss function.

Inference During inference, when the neural language model is utilised for generation, it has no access to the gold input tokens. Therefore a so-called *auto-regressive* generation process is used where the language model estimates the distribution of the next token conditioned *on already generated tokens* $\hat{x}_{1:i-1}$ (Figure 1.2). The choice of output token based on the distribution by the language model is discussed in Section 1.2.4.

1.2.4 Generation Methods

Several problems need to be addressed in order to use a language model for text generation. Firstly, it is not obvious how to choose the length of the decoded sequence. A common method is to generate new tokens until a special token with a meaning "*end of text*" is generated. Secondly, searching for a sequence that exactly maximizes the estimated conditional probability in Equation 1.12 is not feasible as the complexity of searching for all possible sequences with a model with vocabulary size $|V|$ grows exponentially with length $\mathcal{O}(|V|^T)$.

$$
\hat{y}_{1:\hat{T}} = \arg\max_{y_{1:T}} p(y_{1:T}) \tag{1.12}
$$

Therefore, various approximation schemes that trade-off speed and approximation precision are used in practice. We list the ones that we use in our experiments (see Chapter 5).

Greedy decoding In the greedy decoding scheme, a token with highest estimated conditional probability is selected at each step. While simple, this method is suboptimal with respect to Equation 1.12 and it has been shown that this methods leads to generation of sequences that are repetitive [Vijayakumar et al., 2018].

Beam search decoding The beam search decoding [Graves, 2012] addresses the suboptimality of greedy search by keeping B most likely hypotheses at each time step and eventually choosing the hypothesis that has the overall highest probability. A beam search decoding finds sequences with higher estimated probability than greedy decoding and many state-of-the-art methods use it [Edunov et al., 2018, Yang et al., 2019].

Vijayakumar et al. [2018] compared the B hypotheses returned by a beam search decoding and found that these are "nearly identical sequences that differ only slightly from each other." To counteract this Vijayakumar et al. [2018] developed the *diverse beam search*. Here, the hypotheses are split into several groups and receive a penalty if the groups are too similar.

Sampling Holtzman et al. [2020] note that "maximization-based decoding methods such as beam search lead to output text that is bland, incoherent, or gets stuck in repetitive loops." Generation by sampling next token is an effective way of producing more diverse text. Since there is a non-zero probability of generating text that is ungrammatical, several methods of more guided sampling are proposed. Radford et al. [2019] propose the *top-k* sampling method, where a constant $k \in \mathbb{N}$ is selected and the selected token is sampled from the distribution of top-k most probable tokens. Holtzman et al. [2020] propose the *top-p* sampling method, where a constant $p \in (0, 1]$ is selected and the next token is sampled from the smallest subset of vocabulary $V' \subseteq V$ such that $\sum_{t \in V'} p_{LM}(t|x_<) >= p$.

Wolf et al. [2019] use a combination of sampling and beam search, a beam search with sampling, where at each timestep instead of selecting the top-B hypotheses, the B hypotheses are sampled from the estimated sequence distributions.

1.2.5 Conditional Language Modeling

Conditional language modeling refers to estimation of a probability distribution of sequences of tokens $y_{1:M}$ conditioned on input sequence of tokens $x_{1:N}$. If the input sequence is an English source text and the target sequence is its French translation, such a conditional language model can be trained for machine translation [Sutskever et al., 2014]. If the prefix contains preceding dialogue context and the target is a next utterance in the dialogue, the language model can perform dialogue response generation (see Section 1.3).

Encoder-Decoder architecture Sutskever et al. [2014] use an encoder stack of LSTM cells to process the input sequence and keep only the last hidden state of the encoder, which we denote $h_{x,n}$, as a latent representation of the input sequence (Figure 1.2). A decoder stack of LSTM cells estimates conditional probabilities

of target sequences conditioned on the latent representation $h_{x,n}$ (i.e., the initial hidden state of the decoder $h_{y,0}$ is initialized to $h_{x,n}$).

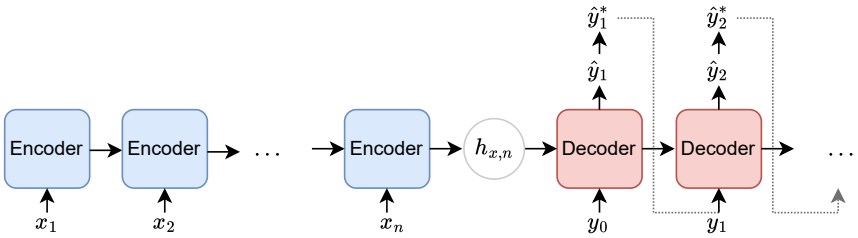


Figure 1.2: Computation of a sequence-to-sequence model. The transformation $\hat{y}_t \rightarrow \hat{y}_t^*$ is guided by a decoding procedure (discussed in Section 1.2.4). Dashed line represents the input choice at the inference time.

Training and inference of the encoder-decoder RNN architecture is conducted in the same way as the training and inference of a plain neural language model (see Sections 1.2.3, 1.2.4), except for not modeling the input sequence.

1.2.6 Attention

In the encoder-decoder architecture discussed in Section 1.2.5, all the information about the source sequence is compressed into a fixed-length latent vector $h_{x,n}$. Bahdanau et al. [2014] note that "this may make it difficult for the neural network to cope with long sentences, especially those that are longer than the sentences in the training corpus."

The proposed solution is to allow the decoder to access all the previous hidden states of the encoder $h_{x,1}, ..., h_{x,n}$ for the update of the hidden vector $h_{y,t} \rightarrow h_{y,t+1}$. A score s_i is computed for each pair $(h_{x,i}, h_{y,t})$ and softmax is used to normalize these scores so that $\sum_{i=1}^{N} s'_i = 1$. A context vector is a sum of encoder hidden states weighted by the normalized scores $c_t = \sum_{i=1}^{N} s'_i h_{x,i}$. The next hidden state $h_{y,t+1}$ is computed as a function of the context vector c_t, current hidden state $h_{y,t}$ and the decoder input y_t. Hence when predicting the token distribution $\hat{y}_{t+1}$, the network has access to all the hidden states from the encoder, effectively removing the bottleneck discussed above.

There are several approaches to computation of the unnormalized score s_i. Bahdanau et al. [2014] use a jointly trained a multi-layer perceptron, while Luong et al. [2015] show that using simple dot product $h_{x,i}^{\top} h_t$ is similarly effective.[2]

1.2.7 Transformer Architecture

The transformer architecture [Vaswani et al., 2017] is the latest major architectural advancement in the field of natural language generation that became mainstream. The model is built on the concept of the self-attention which takes the computational advantage of GPU by parallel processing of a sequence instead

[2]Luong et al. [2015] introduce several other changes that we consider out of scope of this thesis.

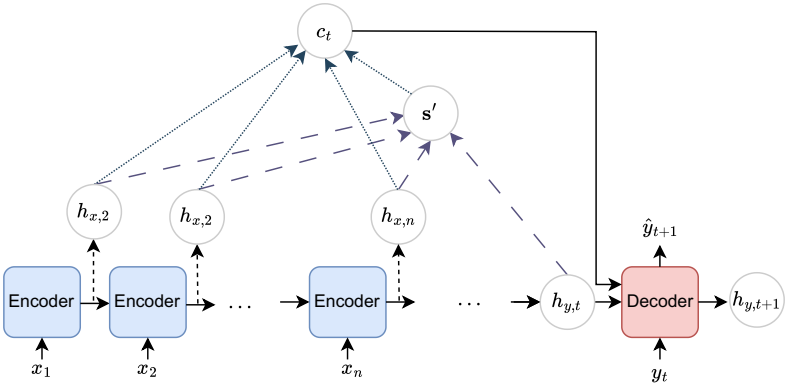


Figure 1.3: Attention model of Bahdanau et al. [2014].

of sequential processing, which is required in RNN architectures. The computational advantage concerns only training and processing of the input sequence, the decoding is still done in auto-regressive way.

In this section we discuss the layers used in the transformer encoder-decoder model. Transformer model is composed of several transformer blocks organized to encoder and decoder stacks. Similarly to RNNs, transformer model operates on sequences of embeddings; however, since transformer blocks do not model position explicitly, a separate positional embedding is added. The hidden representation from the decoder stack is transformed by a linear layer with softmax to produce an estimation of the distribution of the next token.

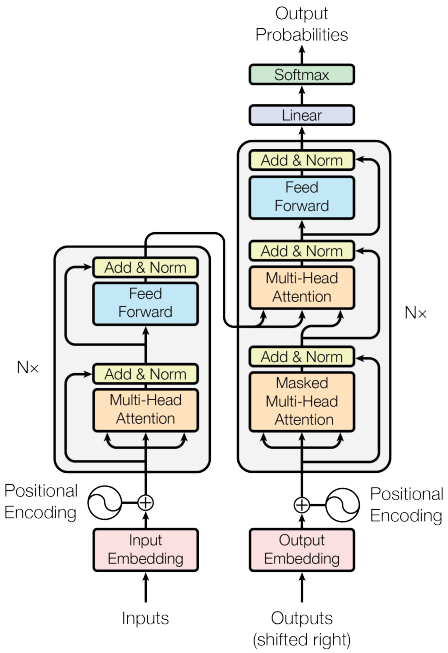


Figure 1.4: Diagram of the transformer architecture. Figure taken from Vaswani et al. [2017]

Positional Encoding While in the recurrent architectures, such as the encoder-decoder with attention (Section 1.2.6), the information about position is im-

plicitly modeled (e.g., hidden state $h_{x,n}$ is computed from $h_{x,n-1}$, hence the network can learn that token x_{n-1} is before token x_n), in transformer blocks, the position information is not taken into consideration.

Since the information is not modeled inside the transformer, the information is added in the form of a positional embedding. The positional embedding e^p is summed with the word embedding e^w to produce the final embedding e used in the transformer model (Equation 1.13).

$$e = e^w + e^p \tag{1.13}$$

Vaswani et al. [2017] propose a positional encoding which statically encodes the information about position (Equation 1.14). However, Gehring et al. [2017] propose to replace these with learned representations (initialized randomly and trained jointly with the rest of the model). This approach was later adopted in all the pretrained models discussed in this thesis (Section 1.4).

$$\begin{aligned} e^p_i &\in \mathbb{R}^{d_{\text{model}}} \\ e^p_{i,2j} &= sin(\frac{i}{10,000^{\frac{2j}{d_{\text{model}}}}}) \\ e^p_{i,2j+1} &= cos(\frac{i}{10,000^{\frac{2j}{d_{\text{model}}}}}) \end{aligned} \tag{1.14}$$

Transformer Block Each encoder block is composed of two sub-layers, multi-head attention and a feed-forward network. Each decoder block is composed of three sub-layers, multi-head attention over the decoder inputs, multi-head cross-attention, and a feed-forward network. Residual connections [He et al., 2016] (the outputs of a layer are summed with its inputs) and layer normalization [Ba et al., 2016] are employed around each sub-layer to stabilize training.

Self-Attention The self-attention layer is at the core of the transformer layer. The inputs of the layer (vectors $x_1, ..., x_N$, $x_i \in \mathbb{R}^{d_{\text{model}}}$) are linearly projected to three different representations, called queries $q_i \in \mathbb{R}^{d_k}$, keys $k_i \in \mathbb{R}^{d_k}$ and values $v_i \in \mathbb{R}^{d_v}$.

Similarly to the dot-product attention (Section 1.2.6), for each query vector q_i a score representing significance of all the key vectors k_j with respect to the query is computed and an updated representation $y_i \in \mathbb{R}^{d_v}$ of the input is produced (Equation 1.15). The dimensionality of the output is different to the dimensionality of the input due to a multi-head attention formulation.

$$\begin{aligned} s_{i,j} &= \frac{q_i^\top k_j}{\sqrt{d_k}} \\ s'_{i,*} &= \text{softmax}(s_{i,1}, ..., s_{i,N}) \\ y_i &= \sum_{j=1}^{N} v_j s'_{i,j} \end{aligned} \tag{1.15}$$

Multi-Head Attention Instead of using a single attention mechanism as was standard for RNNs (Section 1.2.6), h self-attention layers produce output repre-

sentations $y_{i,1}, ..., y_{i,h}$ for each input x_i. These representations are concatenated to a vector $y_{i,*} \in \mathbb{R}^{hd_v}$ and linearly projected to a vector $y_i \in \mathbb{R}^{d_{\text{model}}}$. Vaswani et al. [2017] note that "multi-head attention allows the model to jointly attend to information at different positions, while in a single attention head, averaging inhibits this."

Position-wise Feed-Forward Network A two-layer feed-forward network (Equation 1.16) transforms each input x_i into an output representation y_i.

$$y_i = ReLU(x_i W1 + b_1)W_2 + b_2 \tag{1.16}$$

Masking In the decoder stack of transformer layers masking is used in the self-attention layers in order to prevent the model from attending to "future" positions (to the right of the current one) so that the model can be used for decoding in an auto-regressive manner (see Section 1.2.3).

$$s_{i,j}^{\text{decoder}} = \begin{cases} \frac{q_i^\top k_j}{\sqrt{d_k}} & \text{if } i < j \\ 0, & \text{otherwise} \end{cases} \tag{1.17}$$

Cross-Attention The decoder transformer layer contains one additional cross-attention layer. The computation of this layer is the same as in the standard multi-head self-attention layer. The encoder outputs are transformed to keys and values, while the outputs of previous decoder layer are transformed to queries. The cross-attention mimics the attention in RNN-based encoder-decoder models (Section 1.2.6).

Decoder-Only Models While the original transformer model is based on the encoder-decoder architecture, subsequent work often focuses on the usage of either encoder-only [Devlin et al., 2019, Liu et al., 2019, Conneau et al., 2020], or decoder-only models [Radford and Narasimhan, 2018, Radford et al., 2019, Brown et al., 2020]. The decoder-only model has the same architecture as the encoder part of the original transformer with the auto-regressive masking scheme from the decoder part. It can be used for conditional language generation in the same way as the encoder-decoder transformer by not training with language modeling objective on the input sequences.

1.3 Dialogue Modeling

As described in the introduction, the main task explored in this thesis is the end-to-end dialogue modeling on the ConvAI2 dataset. In this section we provide the necessary definitions for applying neural language models to this task. We loosely follow definitions of Jurafsky and Martin [2024, Chap.-15].

Open-Domain Dialogue A dialogue is a sequence of *turns*. A turn is an utterance contributed by one speaker to the dialogue, hence a turn may consist of one or more sentences (the dialogue in Table 1.1 contains 5 turns). In this text we use term "turn" and term "utterance" interchangeably. Based on the outcome

Who	Turn
A	hello how are you this gorgeous day ?
B	i'm great ! how are you ?
A	fairly good . just trying to decide what i'm going to cook tonight
B	i see . what do you like to eat ?
A	i am a gourmet cook so i make everything

Table 1.1: An excerpt from a dialogue from the ConvAI2 dataset [Dinan et al., 2020]. See Chapter 2 for more details on the data.

of the conversation we distinguish two kinds of dialogue systems. In this thesis, we are concerned with *open-domain*, *non-task-oriented* dialogue, i.e., general social conversation, which can span any possible topic of interest to the two conversation participants (hence open-domain) and does not follow a particular goal, other than entertainment (hence non-task-oriented).

Dialogue Modeling End-to-end dialogue modeling is a direct application of conditional language modeling discussed in Section 1.2.5. An end-to-end dialogue model is trained with a standard language modeling objective (Section 1.2.3), where all preceding utterances $u_1, ..., u_{n-1}$ in the dialogue (dialogue history) are used as context and the model predicts the words from the following utterance auto-regressively. This objective is commonly named *next utterance prediction*.

$$\mathcal{L} = \sum_{i=1}^{|u_n|} (\ln p_{LM}(u_{n,i}|u_{n,1:i}, u_{1:n-1})) \tag{1.18}$$

1.4 Pretrained Language Models

As Weiss et al. [2016] note: "Acquiring large-enough dataset, which captures the data distribution of testing data can be difficult and expensive." Due to expensiveness of acquiring a training dataset matching the distributions of real-world tasks, the paradigm of training deep neural networks slowly changed from random initialization to using transfer learning, or pretraining and finetuning (in the particular sense we are discussing here) [Liu et al., 2023].

Transfer learning can be loosely defined as a technique in which "knowledge learned from a task is re-used in order to boost performance on a related task." [West et al., 2007]. Based on the relation between the source and the target task we distinguish two types of transfer learning.

In *transductive transfer learning*, the source task is the same as the target task; however, the domains are different. As an example, Blitzer et al. [2007] transfer a sentiment classification model trained on book product reviews dataset to DVDs product review dataset.

In *inductive transfer learning* the target task is different from the source task. E.g., in computer vision, the models are pretrained on the image classification task on the ImageNet [Deng et al., 2009] dataset and then finetuned to image segmentation, or object detection tasks on the COCO [Lin et al., 2014] dataset [Girshick, 2015, Qiao et al., 2021].

1.4.1 Transfer Learning in Natural Language Processing

Inductive transfer learning is the most used transfer learning method in the NLP [Liu et al., 2023]. In the beginning, the transfer learning is used as a regularization method to stabilize the training of RNN-based sequence to sequence models [Dai and Le, 2015], later it is shown that pretraining on a large language modeling dataset improves downstream finetuning performance on numerous text classification tasks.

Transfer Learning as a Regularization Method

The training procedure of the seq2seq network [Sutskever et al., 2014] introduced in Section 1.2.5 starts with random initialization of weights in the encoder and the decoder LSTM cells. However, training of the randomly initialized LSTM networks is very unstable [Kolen and Kremer, 2001]. The instability of training made it difficult to increase the capacity of the model (e.g., use deeper stack of LSTM cells, or greater dimension of the hidden vectors), or to increase the number of backpropagation-through-time steps, effectively limiting the modeling of long-range relationships in text. To mitigate the instability of the training of the LSTM-based neural network for sentiment classification, Dai and Le [2015] experiment with two different pretraining objectives.

The sequence autoencoding task (Figure 1.5) [Dai and Le, 2015] consists of recreating the input sequence from the latent vector representation produced by the encoder.

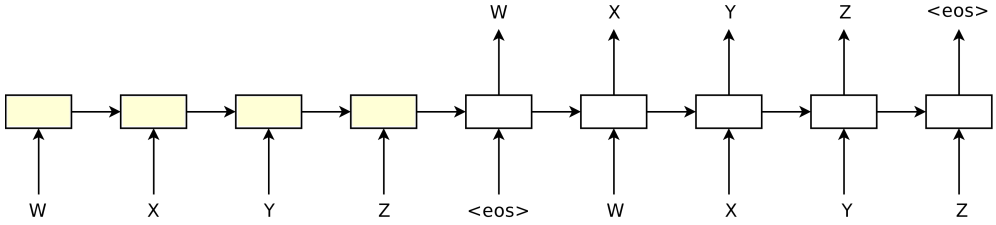


Figure 1.5: Sequence autoencoding task, where the decoder predicts the input of the encoder (encoder is highlighted). Figure taken from Dai and Le [2015].

The language modeling task [Mikolov et al., 2010] consists of predicting the next token based on the previous context (the decoder part of Figure 1.5).

The pretraining is done on the same dataset as the downstream task. E.g., first, the model was pretrained to recreate Rotten Tomatoes movie reviews from the input (sequence autoencoding task) and then the pretrained model was finetuned to predict the sentiment class of a review.

Finetuning of pretrained models proves to be more stable and shows better generalization properties, which allows the practitioners to use deeper architectures and model longer dependencies in the text [Dai and Le, 2015, Howard and Ruder, 2018]. Therefore Dai and Le [2015] were able to use deeper models and model longer dependencies in the text (i.e., they could increase the number of steps in the backpropagation-through-time).

Transfer from a General Domain Dataset

The transfer discussed in the previous section is between two tasks (language modeling / sequence autoencoding and sentiment classification) on the same dataset. Dai and Le [2015] have shown that the transfer is also possible between two different classification tasks and two datasets (language model is pretrained on the Amazon movie reviews sentiment classification dataset [McAuley and Leskovec, 2013] and finetuned for sentiment classification on the Rotten Tomatoes sentiment classification dataset [Pang and Lee, 2005]).

Howard and Ruder [2018] hypothesized that with a general and large enough pretraining dataset, it is possible to transfer to any domain. In their experiment an LSTM-based model is pretrained on the Wikitext-103 dataset [Merity et al., 2017] (about 100 million tokens) with the language modeling objective. This model is then finetuned on several text classification datasets. Howard and Ruder [2018] report that their method has two major benefits. 1) The method "significantly outperforms the state-of-the-art on six text classification tasks." 2) This approach is data-efficient, where the pretrained model requires ten times less data to match the performance of a model trained from scratch on the same dataset.

Howard and Ruder [2018] concluded that a possible future direction is to improve the language model pretraining. Raffel et al. [2020] note that thanks to the abundance of language modeling data on the internet, "better performance is achievable simply by training a larger model on a larger dataset."

1.4.2 Pretrain and Finetune

Radford and Narasimhan [2018] build on the results above. Firstly, a language model with the transformer decoder architecture (Section 1.2.7) is used, which extends the idea that with better language model architectures, the performance on downstream tasks improves. The 124M-parameters pretrained model is named GPT. Moreover, the pretraining is conducted on the BooksCorpus dataset [Zhu et al., 2015] which contains high quality text (about one billion tokens) from unpublished books, thus extending the idea of improvement of downstream results by pretraining on better datasets.

Task Representation While Howard and Ruder [2018] experiment only with the text classification task, the goal of Radford and Narasimhan [2018] is to "learn a universal representation that transfers with little adaptation to a wide range of tasks." Therefore, they reduce all downstream tasks to a setup where a text is fed into the model and the model returns a one dimensional real valued vector. The transformer decoder creates a representation of the text input and then the model's representation of a special `<extract>` token is fed to a linear layer (Figure 1.6). Hence during the finetuning phase, only the linear layer needs to be randomly initialized (as it is not pretrained) and the remainder of the model is initialized from the pretrained weights. With such a setup, the finetuned GPT model reached the contemporary state-of-the-art performance in 9 of 12 studied tasks.

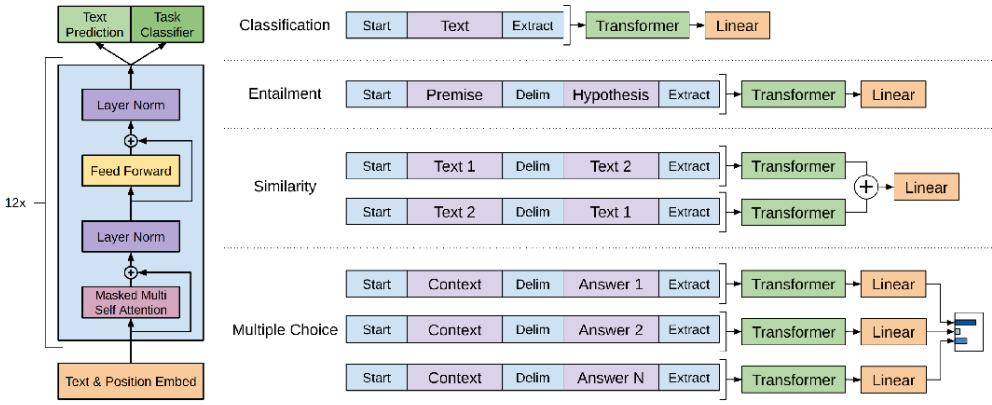


Figure 1.6: Representation of different tasks for finetuning GPT. Figure taken from Radford and Narasimhan [2018]

Zero-Shot Transfer Citing the results above, Radford and Narasimhan [2018] investigated the effectiveness of the language modeling pretraining for transfer to downstream tasks. With the aim of observing the performance of the GPT model in different stages of pretraining, several heuristics were designed to extract task predictions without any change to model parameters. This sort of experiments is called a *zero-shot* experiment [Palatucci et al., 2009], as the model is not finetuned for the task (finetuned on zero examples).

"The performance of these heuristics is stable and steadily increases suggesting that generative pretraining supports the learning of a wide variety of task relevant functionality" [Radford and Narasimhan, 2018]. The experiments also show that the transformer architecture is indeed more performant and more stable (smaller variance in downstream task metrics) than the LSTM architecture (Figure 1.7).

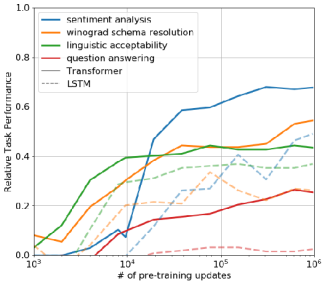


Figure 1.7: Progression of performance on target tasks during pretraining. The solid line depicts the performance of the transformer model, the dashed line depicts the performance of an LSTM model. Figure taken from Radford and Narasimhan [2018].

Library of Pretrained Models Due to large pretraining cost, the paradigm shifts from training a custom task-specific model to pretraining a general fixed architecture on a large language modeling dataset (where it learns general purpose language features), introducing additional task-specific parameters to it and finetuning the model for many different tasks [Liu et al., 2023]. When considering a specific application of a language model, one can inspect a library of pretrained models, such as HuggingFace [Wolf et al., 2020] and choose a pretrained language

models such as GPT (discussed above) [Radford and Narasimhan, 2018], GPT-2 [Radford et al., 2019], BART [Lewis et al., 2020], or T5 [Raffel et al., 2020] and finetune it for the downstream task.

1.4.3 Increasing the Capacity of Language Models

The pretraining of GPT-2 model [Radford et al., 2019] was another extension of the two ideas (increase in model capacity and increase of size of the pretraining dataset) discussed in Section 1.4.1. Firstly, a WebText language modeling dataset was collected, which is about 10 times larger than the BooksCorpus dataset used to pretrain the previous GPT model [Radford and Narasimhan, 2018]. Furthermore, language models with larger capacity in comparison to the original GPT model are used. Four sizes of the model are trained, with 124M (GPT-2-small), 355M (GPT-2-medium), 774M (GPT-2-large), 1558M (GPT-2) parameters.

Zero-Shot Performance The experiments conducted by Radford et al. [2019] focus on the zero-shot downstream performance of the GPT-2 family of models. A task-specific input template (e.g., to use the pretrained language model for question answering task, this input template was used: "Q:<context>A:") is used in order to use a language model for a downstream task (search for a templatized model input with which the model achieves the best downstream performance is called *prompt engineering* and is further discussed in Section 1.5). Similarly to the findings presented in Section 1.4.2, it is shown that over the course of pretraining, the language model improves in solving various downstream tasks.[3] Moreover, with the same templatized input, the downstream performance improves as a function of model size (Figure 1.8).

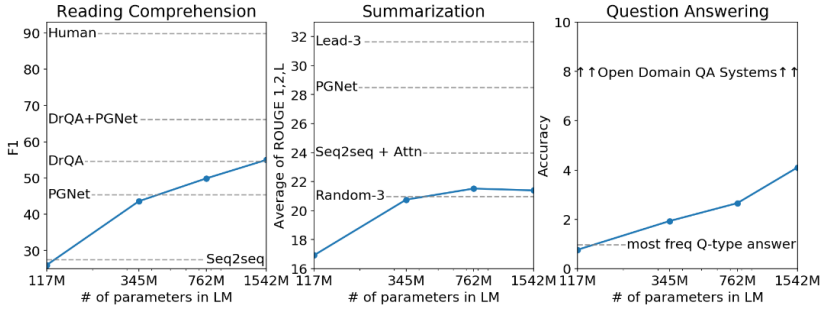


Figure 1.8: The zero-shot performance of GPT-2 family of models as a function of the number of parameters in the model. Figure taken from [Radford et al., 2019].

Effect of Model Size Kaplan et al. [2020] explore that the language modeling performance (measured by the validation cross-entropy loss) depends on the scale of the dataset and the number of model parameters. Moreover the transfer improves with the model size, as larger models are more efficient (fewer optimization steps and smaller task-specific datasets are needed for effective transfer).

[3]The tasks evaluated by Radford et al. [2019] are question answering, summarization of text and reading comprehension. For an introduction to these tasks, we refer the reader to the original paper by Radford and Narasimhan [2018].

The development in subsequent years further focused on increasing the capacity of language models by training models with greater parameter count on greater amount of data (e.g., from 1.5B parameters GPT-2 [Radford et al., 2019] to 8B parameters MegaTron [Shoeybi et al., 2019] to 11B parameters T5-XL [Raffel et al., 2020]).

1.5 Prompting

While the developments in transfer learning discussed above lead to better finetuning performance on many downstream tasks, the language model applicability is still limited by the need of a large labeled in-domain dataset for finetuning (approximately thousands of labeled examples are typically needed for sufficient performance according to Brown et al. [2020]).

In this section we discuss an idea of a general purpose language model that can solve any task formulated in language (Section 1.5.1) and present the GPT-3 model [Brown et al., 2020] that outperforms finetuned state-of-the-art models in a zero-shot setting on several well studied tasks (Section 1.5.2). Next, we explore the *prompt engineering* and *prompt tuning* approaches that leverage the pretrained language model on a downstream task with little to no finetuning (Sections 1.5.3, 1.5.4). Then we discuss instruction tuning and reinforcement learning from human feedback, where the pretrained language models are adapted to follow human instructions more effectively (Sections 1.5.5, 1.5.6).

1.5.1 General Purpose Language Models

First, we illustrate the difference between a general purpose model and a task specific model. A task-specific model is trained to infer the probability $p_{\text{task}}(output|input)$ and hence the input does not need to contain information about the task to be performed. A general purpose model can estimate the probability of the output text conditioned on the input text and a task specification. E.g., for the input text "Who is the American president?" the output text "Asked my professor at the university" should be much more probable for the text continuation prediction task than for the question-answering task.

McCann et al. [2018] show a simple approach to this problem by using a training dataset of triples `(question, context, answer)`, where the seq2seq model is fed with the concatenation of `question` (which contains the task specification) and the `context` and should produce the `answer`. Radford et al. [2019] hypothesize that a large-enough language model can learn to solve any text-to-text[4] task without explicit supervision (i.e., without finetuning the model on a task-specific dataset), as long as the *prompt* (the model input) is constructed in such a way that the model can recognize the task to be solved (see Section 1.4.3).

1.5.2 GPT-3

Brown et al. [2020] follow the ideas of increasing the model capacity and size and improving the quality of the pretraining dataset. The GPT-3 model has

[4] A task where both input and output can be formulated in text.

175B parameters, which is two orders of magnitude larger than the GPT-2 model discussed in Section 1.4.3 and the pretraining dataset consists of 300B tokens, i.e., 50 times more than the WebText dataset used for pretraining GPT-2.

While the zero-shot performance of GPT-2 "is still far from use-able" [Radford et al., 2019], the GPT-3 model shows good results in the zero-shot setting and in the *few-shot* setting (the prompt contains a few demonstrations of task solutions, e.g., for translation task, the prompt may contain several pairs "<English sentence>=<French sentence>") and in the few-shot setting it even achieves state-of-the-art performance on several datasets. In this section we focus on the intuition behind the zero-shot and few-shot model performance, the downstream performance of the model and the implications of using models with more than 100B parameters.

Intuition Behind Zero-Shot and Few-Shot Performance

Brown et al. [2020] present a simple intuition behind the zero-shot and few-shot prompting. In the zero-shot prompting (i.e., the prompt contains only the natural language description of the task and the task input), the model should recognize a pattern that was already seen during training (e.g., adding the prefix `TL;DR:` induces a summarization behavior). In the few-shot prompting, the model should recognize a question-response pattern in the input and continue the pattern.

Implications

Based on the empirical findings by Brown et al. [2020], a new way of using language models emerged. Instead of the transfer learning approach (pretrain and finetune), one can easily use a pretrained language model by simply instructing it with a prompt, e.g., "`Correct grammar in this sentence: <english sentence>:`"

However, due to the number of GPT-3 model parameters (175B), the hard drive requirements to store weights (about 700GB for full 32-bit float weights) are so large that finetuning the model for many downstream tasks is infeasible. Similarly, as of June 2024, there is still no single GPU with a large-enough memory to run even the inference of the model. Therefore several model parallelism strategies [Shoeybi et al., 2019] need to be employed to distribute the computation onto several GPUs.

To conclude, while the GPT-3 model is a display of great zero-shot and few-shot capabilities, due to its sheer size, the infrastructure cost associated with deploying it is so large that models of this size are deployed only by a few large companies. Nowadays, several smaller models display zero-shot and few-shot capabilities exceeding the GPT-3 performance [Touvron et al., 2023, Bai et al., 2023]; however, in order to deploy them on a single GPU, advanced model quantization techniques are still needed [Dettmers et al., 2022a, Wu et al., 2023].

1.5.3 Prompt Engineering

As the new wave of language models is able to follow instructions, naturally one wants to find the instruction that maximizes the performance on the given task. Wei et al. [2022] use 10 different prompts on several common benchmarks and

show that the best prompt on a held-out development dataset can improve the test set zero-shot performance of the LAMDA-PT model [Thoppilan et al., 2022] with 137B parameters by up to 5%.

Prompt engineering is an optimization method aiming to find a templatized input that results in the most effective performance of the model on a downstream task. While manual search of prompts is possible, Liu et al. [2023] cite several automatic discrete prompt search methods such as mining examples similar to the task from the pretraining dataset, paraphrasing, or generating a prompt with another (presumably smaller) language model.

1.5.4 Prompt Tuning

The prompt engineering approach presented above has a big downside; it is not a differentiable optimization technique. Assuming that there exists a task prefix, i.e., a sequence of discrete tokens that improves model's performance on a task, which can be found by prompt engineering, we may relax the discreteness constraint and search for the so called *soft tokens*. Lester et al. [2021] and Li and Liang [2021] add *soft tokens* to the embedding layer, or to all transformer layers of the model and use a training regime where the remainder of the model is frozen and only the *soft tokens* are optimized.

Prompt tuning techniques make it possible to improve the performance of the model on a specific downstream task while keeping all the pretrained weights frozen. Effectively, the problem with storage requirements outlined in Section 1.5.2, is fought as only a single copy of the pretrained model and a small set of task-specific finetuned parameters has to be stored.

Prompt tuning techniques are part of a large family of techniques called PEFT (parameter efficient finetuning), where only a small fraction of pretrained model weights is finetuned on the downstream task [Hu et al., 2022, Dettmers et al., 2023].

1.5.5 Instruction Tuning

Another line of improvements of the pretraining procedure is the *instruction tuning*, where the language model is explicitly finetuned to follow instructions.

Wei et al. [2022] take a pretrained 137B language model LAMDA-PT [Thoppilan et al., 2022] and finetune it on a combination of 60 datasets which are described via instructions. Finetuning on this combined dataset "substantially improves zero-shot performance on unseen tasks," beating even the few-shot performance of the GPT-3 model [Wei et al., 2022].

It is shown that the zero-shot performance on unseen tasks improves with the number of tasks on which the model is finetuned and with the size of the model [Chung et al., 2024, Longpre et al., 2023].

1.5.6 Reinforcement Learning from Human Feedback

Pretrained and instruction tuned language models show an ability to solve tasks described by instructions. However, these models may still produce generations with unwanted properties, which can pose risks when deployed to uncontrolled

environments. Reinforcement learning from human feedback (RLHF) [Ouyang et al., 2022] is another step in improving the zero-shot prompting performance of language models that partially mitigates the probability of generation of toxic or biased content.

After pretraining on a huge language modeling dataset and finetuning on a large instruction tuning dataset, Ouyang et al. [2022] add another stage, where the model is aligned to human preferences through reinforcement learning.[5]

Risks Posed by Language Models

Lucy and Bamman [2021] show that when the GPT-3 model is prompted to create a fictional story, "feminine characters are more likely to be discussed in topics related to family, emotions and body parts, while masculine ones are more aligned to politics, war, sports and crime." These implicit gender and sociopolitical biases may cause harm to humans, as e.g., model prompted to analyse a CV may be less likely to recommend historically discriminated groups to recruiters [Weidinger et al., 2022]. Ouyang et al. [2022] try to mitigate the risk of producing such utterances by finetuning the language models to produce more truthful text, without being "biased, toxic, or otherwise harmful."

Training Procedure

The RLHF procedure is conducted on a pretrained and instruction-tuned base model. A dataset of generations by the base model is sampled and human evaluators rank these generations considering criteria such as *truthfulness*, *helpfulness* and *harmlessness*. A copy of the base model that is called a reward model is trained in the same way as pairwise ranking models discussed in Section 1.6.2. The reward model is used as a proxy to human preferences during the reinforcement learning stage.

Another copy of the base model, which we call the RLHF model, is then trained with the proximal policy optimization algorithm [Schulman et al., 2017]. The reward, which specifies the weights update of the RLHF model, is composed of the score assigned by the reward model and of the KL-divergence between the distributions estimated by the RLHF model and the base model. The KL-divergence should encourage the RLHF model to explore new regions instead of collapsing to a single mode and to not learn to produce outputs that are too different from those that the reward model has seen during training [Stiennon et al., 2020].

Results

The generations by the RLHF model, are preferred by human evaluators considering the same criteria as above in comparison to outputs from either only pretrained, or pretrained and instruction-tuned model.

[5]For an introduction to reinforcement learning we refer the reader to Sutton and Barto [2018]

GPT-3.5, GPT-4, GPT-4o

The large-scale human evaluation benchmark ChatBot Arena [Chiang et al., 2024] (further discussed in Section 3.2) suggests that proprietary models GPT-4 and GPT-4o are among the most capable language models, while GPT-3.5 is more price-effective and still ranks in the upper half of their leaderboard. These models are trained with the RLHF; however, due to their proprietary nature, there is no official information on neither the architecture nor the number of model parameters, or a training dataset. We experiment with these models in Section 5.6.

1.5.7 Direct Preference Optimization

Rafailov et al. [2023] note that "RLHF is a complex and often unstable procedure," and propose a "new parametrization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form."

Instead of using reinforcement learning, they propose to align the language model to human preferences by optimizing a specially crafted binary cross-entropy loss (Equation 1.19) on the same dataset that is used to train the reward model in the RLHF setup. During training, the gradient of the loss increases the likelihood of the preferred completions y_w and decreases the likelihood of the dispreferred completions y_l, while accounting for the KL-divergence.

$$\mathcal{L}_{\text{DPO}}(\text{RL};\text{SFT}) = -\mathop{\mathbb{E}}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\beta\log\frac{\text{RL}(y_w|x)}{\text{SFT}(y_w|x)} - \beta\log\frac{\text{RL}(y_l|x)}{\text{SFT}(y_l|x)}\right]. \quad (1.19)$$

1.6 Learning to Rank

In the ranking problem, we consider a dataset of pairs $(\mathbf{x}, \mathbf{y})$, where $\mathbf{y} \in \mathbb{N}^n$ is the ranking of candidates $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$ and the task is to train a model that would learn to reproduce this ranking [Fürnkranz and Hüllermeier, 2010, Chap.-1].

Ranking cannot be directly modeled with neural networks since sorting is not a differentiable operation. We explore two methods of training of a neural ranking model, namely pointwise ranking (Section 1.6.1) and pairwise ranking (Section 1.6.2).[6] In our experiments (see Section 5.4), we train ranking models for a two-stage dialogue models. In the first stage a generation model generates several response candidates and in the second stage the candidate ranked highest by a ranking model is selected as the final response.

1.6.1 Pointwise Ranking

Pointwise ranking [Caruana et al., 1995] is implementation-wise the simplest form of ranking, which transforms the ranking problem into a regression problem. The dataset of rankings is transformed to a dataset of pairs (x, y), where x is the

[6]There is a vast array of other possibilities that we do not discuss nor experiment with, such as gradient-boosting ranking [Burges, 2010], list-wise methods [Cao et al., 2007], or soft ranking [Taylor et al., 2008]. Our main reasons for not selecting these methods are the ease of experiments and implementation.

candidate and y is its score. The ranking model r is trained with mean squared error loss to estimate a score of input x which is close to the gold score y (Equation 1.20).

$$\mathcal{L} = \sum_{(x,y)\in\mathcal{D}} (y - r(x))^2 \tag{1.20}$$

The main difficulty of this approach lies in transforming a dataset of rankings into a dataset of artificial scores.

1.6.2 Pairwise Ranking

In contrast to pointwise ranking, the main idea of pairwise ranking is to model the ranking problem as a binary classification problem, where the task is to choose which candidate out of a pair should be ranked higher without explicitly providing scores of candidates. Hence, the difficulty of an artificial transformation from a dataset of rankings to a regression dataset discussed above is overcome. However, the raw dataset of rankings still has to be transformed to a dataset, where each sample consists of a pair of candidates (x_1, x_2) and their respective ranks (y_1, y_2).

We experiment with two different training schemes for the pairwise ranker which are inspired by RankNet [Burges et al., 2005] and LambdaRank [Burges et al., 2006].

Pairwise Ranking as Binary Classification

In both pairwise training schemes, during training the ranking model is fed with two inputs, (x_1, x_2), which are associated with target ranks (y_1, y_2). If the first input is higher in the ranking ($y_1 < y_2$), then the ranking model r should assign it higher score ($r(x_1) > r(x_2)$). It follows that if we treat the indicator $[\![y_1 - y_2 \leq 0]\!]$ as a label and difference $(r(x_1) - r(x_2))$ as a model prediction, we can model the ranking problem as a binary classification over pairs of inputs. Binary cross entropy (Equation 1.21) is a natural choice of loss function for this task.

$$\begin{aligned} \hat{s} &= \text{sigmoid}(r(x_1) - r(x_2)) \\ s &= [\![y_1 - y_2 \leq 0]\!] \\ \mathcal{L}_{\text{RankNet}} &= -s \log(\hat{s}) - (1 - s) \log(1 - \hat{s}) \end{aligned} \tag{1.21}$$

RankNet

RankNet [Burges et al., 2005] is trained exactly in the way explained above. During training, pairs of inputs are fed into the network and binary cross-entropy is used as a loss function. During inference, raw model scores are predicted for individual candidates, then the candidates are simply ranked according to the predicted scores, with the top-scoring one coming first.

We note that in the reinforcement learning from human feedback (discussed in Section 1.5.6) the reward models are trained with the same pairwise ranking loss from Equation 1.21.

The main deficiency of the RankNet optimization scheme is that it directly optimizes the pairwise correct metric (percentage of correctly ordered pairs) and

the full ranking is only optimized indirectly. We illustrate the deficiency on an artificial example. Suppose there are two pairs of incorrectly ordered input candidates (x_{1_1}, x_{2_1}), (x_{1_2}, x_{2_2}) and their corresponding orders are (2, 3), (1, 8). Intuitively, the loss should punish the model much more for making an error in ranking of the second pair. However, the RankNet training loss punishes both pairs exactly the same way.

LambdaRank

Intuitively, errors in pairs where the difference of target ranks is greater should produce a stronger optimization signal. Burges et al. [2006] introduce LambdaRank which is an approach that aims to optimize more effectively the performance of a ranking model in any metric over rankings and relies on the same intuition.

In the LambdaRank optimization scheme, the RankNet loss of each input pair is multiplied by a scaling factor, which is greater if reordering of the pair causes greater difference in target metric (Equation 1.22).

$$\mathcal{L}_{\text{LambdaRank}} = -|\Delta\text{switch}(x_1, x_2)|\mathcal{L}_{\text{RankNet}} \tag{1.22}$$

2. ConvAI2 Dataset

In this thesis, we focus on open-domain dialogue modeling. We have chosen the ConvAI2 dataset [Zhang et al., 2018] for end-to-end dialogue model training as it is a popular benchmark on non-task-oriented open-domain chat, with a specific accent on maintaining consistent personality, which is one of this thesis aims. This chapter begins with a list of other open-domain dialogue modeling datasets (Section 2.1). We follow with an introduction to the ConvAI2 dataset (Section 2.2). We discuss the dialogue collection (Section 2.3) and the dataset statistics (Section 2.4). Lastly, we manually analyze the dataset and discuss some issues in the dataset samples (Section 2.5).

2.1 Open-Domain Dialogue Datasets

To train open-domain conversational systems, various datasets have been collected, each focusing on different aspect of human conversation. To provide further context for our own work, we introduce some of the most influential open-domain dialogue datasets.

Empathetic Dialogues The Empathetic Dialogues dataset [Rashkin et al., 2019] is composed of triples (emotion, situation, dialogue). In each dialogue, one of the speakers describes the situation while expressing an emotion, while the other speaker, called *a listener* responds in empathetic way. The dialogue model is trained for the listener role and it should implicitly learn to respond in an empathetic way.

Wizards of Wikipedia Dinan et al. [2019] focus on knowledge grounding of open-domain dialogue. The Wizards of Wikipedia is a crowdsourced dataset composed of triples (topic, knowledge, dialogue). The interlocutors discuss a topic. One of the interlocutors is called a wizard and has access to the knowledge - a set of sentences from Wikipedia articles - The other interlocutor is called an apprentice and they play a role of "a curious learner, eager to chat". The dialogue model is trained for the wizard role, where the model should use the information in the knowledge to respond to the apprentice.

BlendedSkillTalk Responding in an engaging way, empathetically, with a consistent persona and using knowledge are all desirable properties of an open-domain dialogue model. Smith et al. [2020] note that these properties are modeled on separate datasets and "a good open-domain conversational agent should be able to seamlessly blend them into one cohesive conversational flow." In order to train a model to blend the aforementioned skills, the BlendedSkillTalk dataset is collected, where each sample contains a dialogue, persona description and a Wizards-of-Wikipedia-like unstructured knowledge text. In the dialogues, the interlocutors should maintain a consistent persona and be empathetic, while grounding their responses in the provided knowledge.

Dialogue	Utterances per Dialogue
ConvAI2	14.7
Wizards of Wikipedia	9.0
Empathetic Dialogues	2.6
BlendedSkillTalk	11.2
Multi-Session Chat	43.0

Table 2.1: Average number of utterances in the discussed datasets.

Multi-Session Chat Dialogues from all datasets mentioned above are relatively short (Table 2.1). Xu et al. [2022] cite this as a major shortcoming and collect a dataset, where crowd workers talk in long dialogues spanning over multiple sessions. A session is a sub-dialogue consisting of at most 14 utterances. After a number of hours, the crowd workers are paired again, while having access to all the previous sessions, i.e., they may address knowledge from past sessions, which creates dialogues, which are on average three times longer than dialogues from the ConvAI2 dataset.

2.2 ConvAI2 Dataset

ConvAI2 dataset [Zhang et al., 2018] contains multi-turn conversations, each between two humans. The interlocutors are provided with a character description composed of 3 to 5 sentences (these descriptions are called *personas*) and are asked to play this character in the conversation. In the dialogues, the interlocutors discuss their characters and try to find out information about the other person (Table 2.2).

2.2.1 Motivation

Zhang et al. [2018] note that at the time of collection of the dataset, the main problems of sequence-to-sequence generative dialogue models were lack of specificity (for instance, instead of answering the question or providing an opinion, the bot simply states "I don't know.") and lack of consistent personality. (for instance, at the start of the conversation, the bot is an avid reader while at the end it does has never read a single book). Similar cases of inconsistencies have previously been demonstrated by Li et al. [2016c] on a conversation between a human and a contemporary state-of-the-art dialogue model, where the model answers the same question with three contradictory responses (Table 2.3).

Zhang et al. [2018] create the PersonaChat dataset to tackle both of these issues by including profile information. In Table 2.4 we can see an excerpt from a sample from the dataset. A model trained on this sample should learn to use the relevant information from the persona. E.g., when generating the third utterance "cool, i am currently studying and enjoy going fishing in my spare time ," the model should learn to reuse the information from its persona.

(a) Persona A

No.	Trait
1	i'm a christian.
2	my father used to work for microsoft.
3	my favourite band is kings of leon.
4	i work in the it industry.

(b) Persona B

No.	Trait
1	i go visit them sometimes.
2	my mother is a social worker for the school system.
3	my best friend lives in a big city.
4	i work at a grocery store.
5	she likes to grow plants in her garden.

(c) Dialogue

Who	Turn
A	good morning . how is everything ?
B	hello there . it is good . i just had work at a grocery
A	sales is difficult in my opinion . i . t . is what i do .
B	i love gardening . and growing plants in the garden
A	bless your heart ! flowers are beautiful
B	thank you . just like living in a big city
A	i live off of the bible .
B	oooh . that's good . have you eaten it
A	why would i eat the old testament ? that's kinky
B	maybe . it tastes good . or well . bread
A	wut ? lol changing the topic . kings of leon is an awesome band
B	oh yea ? i visit them sometimes

Table 2.2: Sample 4528 from the training split of the dataset.

Who	Turn
User	Where do you live now ?
Model	I live in Los Angeles.
User	In which city do you live now ?
Model	I live in Madrid.
User	In which country do you live now ?
Model	England, you ?

Table 2.3: Inconsistent responses generated by a 4-layer Seq2Seq LSTM model. Conversation snippet taken from Li et al. [2016c]

No.	Trait
...	...
5	i watch foreign movies and tv shows.

(a) Persona B

No.	Trait
...	...
3	i am a student.
4	i like to go fishing.

(b) Persona A

Who	Turn
A	hi , tell me a little about yourself !
B	hi i love watching spanish movies and tv shows
A	cool , i am currently studying and enjoy going fishing in my spare time .
...	...

Table 2.4: Sample 4564 of the training split of the ConvAI2 dataset.

2.2.2 Dataset Versions

There are two official versions of the dataset. The PersonaChat dataset [Zhang et al., 2018][1] and the ConvAI2 dataset [Dinan et al., 2020].[2] The ConvAI2 dataset is a preprocessed version of the PersonaChat dataset, used in the ConvAI2 shared task competition.[3] From our analysis, the preprocessing only includes contracting long verb forms ("I have" → "I've") and no data filtering is applied (the number of dialogues and utterances, as well as persona descriptions is the same in both datasets). Moreover, the ConvAI2 dataset does not contain the test split, which was made private for the competition and was not released afterward.

We use the ConvAI2 dataset so that our measured results are comparable to related works (see Chapter 4) which report their evaluation on the ConvAI2 version of the dataset, in line with the previous competition. As automatic metrics used in the evaluation (see Chapter 3) are sensitive to subtle preprocessing, metrics measured on the PersonaChat version of the dataset are not comparable.

[1]The PersonaChat dataset can be downloaded from `http://parl.ai/downloads/personachat/personachat.tgz` (accessed: 07/07/2024)

[2]The ConvAI2 dataset can be downloaded from `http://parl.ai/downloads/convai2/convai2_fix_723.tgz` (accessed: 07/07/2024)

[3]`http://convai.io/2018/`

2.3 Data Collection

The dataset collection process by Zhang et al. [2018] ran in two stages on Amazon Mechanical Turk [Crowston, 2012]. In the first stage, the crowd workers were asked to create a character description, consisting of 5 sentences, at most 15 words each. They were specifically told to invent fictional information and not to include any personal information about themselves. An example persona collected in the first stage is displayed in Table 2.5.

I am a vegetarian.
I like swimming.
My father used to work for Ford.
My favorite band is Maroon5.
I got a new job last month, which is about advertising design.

Table 2.5: Example persona provided to the crowd workers.

In the second stage, the crowd workers were randomly paired, each crowd worker was provided with a persona and asked to "chat with the other person naturally and try to get to know each other" [Zhang et al., 2018].

Zhang et al. [2018] noticed that the crowd workers tend to talk only about the given character description and additionally instructed them to "Both ask questions and answer questions of your chat partner." The crowd workers were prevented from copying from their persona by a simple regular-expression-based filtering implemented in the crowdsourcing chat environment.

For each true utterance in the dataset, a set of additional 19 negative candidates is included as a mean of evaluating retrieval based dialogue models (these models do not generate new text but pick the best response from a set of candidates [Lowe et al., 2015]). These candidates are randomly sampled from other dialogues in the training dataset.

2.4 Statistics of the PersonaChat Dataset

The ConvAI2 dataset contains 9,939 dialogues composed of 147,040 utterances, divided into the training and validation splits.

Split	# Dialogues	# Utterances
Training	8,939	131,438
Validation	1,000	15,602

Table 2.6: Number of samples in dataset splits

2.4.1 Personas

Zhang et al. [2018] state that during the crowd sourcing of the dataset, a set of 1055 personas was collected, 955 of which are used for the training split of the dataset and a hundred for the validation split.

After several preprocessing steps, working with the ConvAI2 version of [Dinan et al., 2020] (we remeasured the results on the PersonaChat version of the dataset,

and found out that these two versions of the dataset do not differ in this regard), we arrive at almost the same numbers, counting 956 different personas for the training set and a hundred for the validation set. In this section, we look at the differences between the statistics measured on the ConvAI2 dataset and the statistics published by Zhang et al. [2018].

Length of Personas Contrary to the claim that each persona consists of five sentences, personas in the available dataset contain any number between three to five profile sentences (Figure 2.1). There are 5962 distinct sets of persona descriptions in the training set, 541 in the validation set, but they are based on overlapping subsets of five sentences. To get to the same statistics as reported in the original paper, we applied the preprocessing steps listed below.

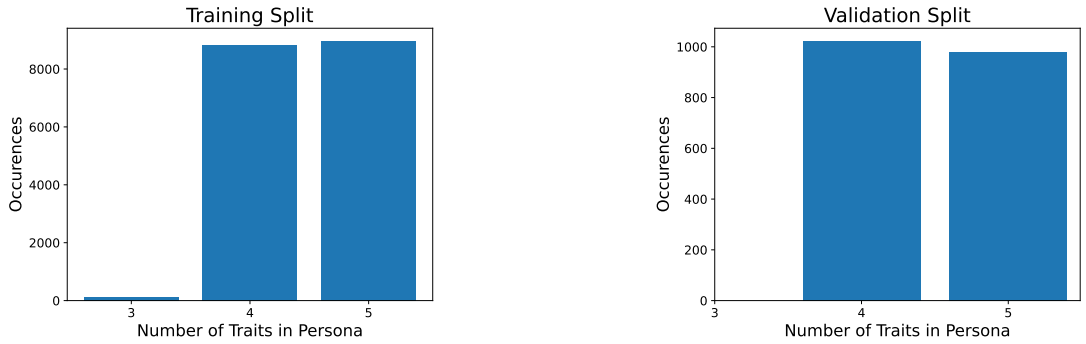


Figure 2.1: Total number of personas of specified length, where each occurrence of persona in the dataset is counted separately.

Persona Subsets We counted all the subsets of a longer persona as one occurrence (see Table 2.7).

No.	Trait
1	i work for an architect firm.
2	i love to cook.
3	i am looking for someone.
4	i have three dogs.
5	i enjoy fishing.

(a) Sample 2226.

No.	Trait
1	i enjoy fishing.
2	i work for an architect firm.
3	i am looking for someone.
4	i love to cook.

(b) Sample 2227.

Table 2.7: Two persona sets from the training split of the dataset. To get to the same statistics as reported by Zhang et al. [2018], we count both of these personas as a single unique instance.

Typographical Differences If two personas differ only by one of these differences in the character description sentences, we count them as a unique instance:

1. contractions (its/it is/it's)
2. typographical errors (participaed/participated)
3. typographical errors in proper nouns (nightfish/nightwish)
4. possesives ("lawyer s office"/"lawyers office"/"lawyer's office")

We used regular expressions to correct for these kinds of differences and after the corrections we count 956 unique personas in the training dataset, a number which is almost the same as the number reported by the authors (955). In our experiments in Chapter 5 we use the original version of the dataset.

2.4.2 Dialogues

Automatic metrics that are commonly reported on this dataset are computed at the word-level, hence an analysis of word level statistics may give us insight on what the models could learn on the dataset. We use the official ConvAI2 tokenizer for counting words (the tokenizer specifics are further discussed in Section 3.1.1).

Statistic	Training	Validation
Mean Word Length	9.64	9.83
Median Word Length	9	10
Word Entropy	6.12	6.03
Percentage of Questions	31.27%	30.09%

Table 2.8: Word-Level Dataset Statistics

The distributions of lengths of the training and validation splits are similar (Table 2.8). It is insightful to look at the examples with minimum and maximum number of words. The example with the minimum number of words (Table 2.9) contains a dialogue with a bizarre situation, where person A does not know what they do for living. In the example with the maximum number of words (Table 2.10) person B is overly specific.

Who	Turn
B	what do you do for a living ?
A	not sure .
B	? ?
A	what about you ?
B	you do not know what you do ?

Table 2.9: Sample 8395 from the training split. Highlighted row contains the shortest utterance in the dataset with 0 words as counted by the ConvAI2 tokenizer [Dinan et al., 2020].

Who	Turn
A	hi , can i paint you ?
B	sure ! but do not paint me with a cat , i hate cats .
A	why ? i was going to paint you with taylor swift .
B	i do not like taylor swift , my dad is always listening to her music and i do not get along with my dad .

Table 2.10: Sample 240 from the validation split. Highlighted row contains the longest utterance in the dataset with 23 words.

2.5 Data Quality

During our experiments, we examined hundreds of randomly picked samples from the training split of the dataset. This made it apparent to us that the data has some quality issues that may make tranining models more challenging. We list the issues below, with examples.

Ordering of Personas It is clear that personas were written sequentially, i.e., individual sentences do not make sense without context. However, sometimes personas in the dataset come in erroneous order (Table 2.11).

No.	Trait
4	i work at a grocery store.
2	my mother is a social worker for the school system.
5	she likes to grow plants in her garden.
3	my best friend lives in a big city.
1	i go visit them sometimes.

Table 2.11: Persona from sample 4528 from the training split of the dataset. The values in the first column display the original ordering of the persona

Grounding Common ground refers to the set of facts known to both interlocutors in the dialogue that they agree on [Jurafsky and Martin, 2024]. Some of conversations from the dataset are entirely based on outside-world events. E.g., in Table 2.12 the interlocutors discuss events in Florida. Without the context (in 2017, during the data collection, there was hurricane Irma in Florida), sentences like "living in florida so it is difficult not to worry ." do not make much sense.

Who	Turn
A	good afternoon , how are you today ?
B	it is a good afternoon . i am fine , how bout yourself ?
A	ok , just worried about this hurricane .
B	ya it is very scary . but god is in control
A	true . living in florida so it is difficult not to worry .
B	oh no ! i am so sorry . are you evacuating ?
A	no , we aren't in an evacuation zone yet so we are ok
B	that's good . hope you and your family are ok
A	we should be . just worried about power outages . i love to cook good meals .

Table 2.12: Sample 890 from the train dataset, where interlocutors make a reference to a hurricane in Florida, which is not mentioned anywhere in the context.

Situations when the crowd worker is asked about preferences or personal background that is not mentioned in the persona is another instance of introduction of not grounded responses to the dataset dialogues. In these situations crowd workers face a dilemma whether to come up with some ungrounded fact, or respond unspecifically. We have already seen an example of an unspecific response which completely confused the conversation partner (Table 2.9).

In Table 2.13 the crowd worker chooses to come up with ungrounded fact (neither the information about food preferences nor music preferences is in the persona description of person A).

Who	Turn
B	that sounds fun ! what would you have for dinner ? i had italian . i love it !
A	i was in the mood for some hamburgers tonight
B	yum ! who your favorite band ? mine is iron maiden .
A	i actually have been listening to old jazz lately . whats your favorite music ?

Table 2.13: Sample 3279 from the training dataset, where person A answers with completely unfounded facts.

Since the average length of a dialogue is only 15 turns (Table 2.1), the dialogue does not contain opportunities to further discuss mentioned facts. Hence the objective of making open domain dialogue systems more consistent (Section 2.2.1) may not be fully achieved.

Language Quality The dialogue collection was conducted in a social-network-like conversation setting. The crowd workers used an informal language common to social networks (Figure 2.2).

> *whats twats the name an location* of your favorite place ?
> oh *you married* ? you farm ? *i love swimming have since a child*
> i am waiting *on* my personal trainer to arrive we are going to ride bikes

Figure 2.2: Ungrammatical social-network-like utterances from the ConvAI2 dataset.

Unsafe Data No safety measures were put in place during the dataset collection. We list two issues that we found during manual analysis of the dialogues in the dataset: 1) harmful advice (Tables 2.14, 2.15), and 2) aggressive language (Table 2.16)

Who	Turn
A	i am a middle school teacher for 8th grade students .
...	...
B	oh sounds fun . i heard that kids behave well when you give them sedatives .

Table 2.14: Sample 2873 from the training split of the dataset, where person B advises middle school teacher to give sedatives to their kids.

World Knowledge The facts mentioned in the persona should serve as conversation starters. However, if a crowd worker happens to not know any additional information about these facts and if their conversation partner asks further, they

Who	Turn
B	nice ! i have never done drugs before
A	drugs are the best . i highly recommend advil
B	i do not know where you buy drugs at
A	walgreenshas the best prices . and a good rewards card
B	i think there is one nearby me !

Table 2.15: Sample 830 of the training split of the dataset, where person A recommends drug buying options.

Who	Turn
A	i have two dogs now , they are better than most people .
B	not better than my two sons . meet me outside how about that ?

Table 2.16: Sample 7914 of the training split of the dataset, where person B wants to start a fight.

have to reveal that they do not know anything about their personality (Table 2.17).

Who	Turn
B	funny ! do you like music ? the muggers is my favorite band
...	...
A	what kind of music is muggers ?
B	it is the assigned character to the left haha i have no idea

Table 2.17: Sample 5468 from the training split of the dataset, where person A reveals that they are chatting in the Amazon Mechanical Turk environment.

3. Metrics

In this chapter, we discuss metrics that assess the quality of open-domain dialogue systems. Automatic metrics discussed in Section 3.1 can be computed fast and without any explicit supervision; however, their effectiveness in comparing different dialogue models is limited. On the other hand human evaluation of a dialogue model (Section 3.2) is our target measure of model quality, but preparing and conducting it is difficult, time-consuming and expensive. Both methods are therefore combined in practice. Automatic evaluation is favored for model development, but the final product is verified using human evaluation.

3.1 Automatic Metrics

We discuss three sets of automatic metrics in this section. The first set are metrics which were selected by Dinan et al. [2020] to evaluate the performance of models on the ConvAI2 dataset, namely perplexity per token (Section 3.1.1), word-level F1 score (Section 3.1.2) and hits@1 on next utterance classification (Section 3.1.3). We adopt all the ConvAI2 metrics for this thesis. The second set are additional metrics that we use in this thesis to explain the generation performance of the model (Section 3.1.4). Lastly, to provide a broader context, we discuss other open-domain dialogue metrics that were not used (Section 3.1.5).

3.1.1 Perplexity per Token

Language models are trained by minimizing the cross-entropy loss H between a target one-hot distribution of tokens $p^*(x_t|x_{1:t-1})$ and the estimated distribution of tokens $p_{LM}(x_t|x_{1:t-1})$ (as discussed in Section 1.2.3). The perplexity per token [Jelinek et al., 1977] of a probability model refers to exponentiated cross-entropy between a data distribution and an estimated distribution by a language model (Equation 3.1).

$$\begin{aligned} H(p^*, p_{LM}) &= \frac{1}{N}\sum_{i=1}^{N}\sum_{x\in V} p^*(x|x_{1:i-1}) \ln p_{LM}(x|x_{1:i-1}) \\ &= \frac{1}{N}\sum_{i=1}^{N} \ln p_{LM}(x_i|x_{1:i-1}) \end{aligned} \tag{3.1}$$

Hence, it is a metric that is directly optimized during the training of a language model. Intuitively, if a language model has perplexity per token P, it can be thought of as selecting uniformly and independently among P possibilities for each token.

$$P(p_{LM}) := e^{H(p^*, p_{LM})} \tag{3.2}$$

However, perplexity per token cannot be used for comparison of two models which use different tokenizers. We show the drawbacks of such an approach on an artificial example by Sabrina J. Mielke.[1] We consider two different subword tokenizations of the phrase "the deforestation," t_1 = [`the`, `de@@`, `forest@@`, `ation`,

[1] `https://sjmielke.com/comparing-perplexities.htm`; accessed: 06/07/2024.

<EOS>] and t_2 = [the, defor@@, estation, <EOS>] and two corresponding language models trained with these tokenizations. Let us assume that the per-token perplexities indicate that the model trained with tokenization t_1 is better.

$$\begin{aligned} ppl^{t_1} &= 19 \\ ppl^{t_2} &= 24 \end{aligned} \tag{3.3}$$

We compute the per-token negative log likelihood by taking natural logarithm of the per-token perplexity. Afterward we multiply the per-token negative log likelihood by the number of tokens to get phrase-level negative log likelihood. We see that the phrase-level negative log likelihood H_1 is higher than H_2, i.e., the ordering of models is the opposite.

$$\begin{aligned} H_1 &= \ln ppl^{t_1} \cdot 5 \approx 14.7 \\ H_2 &= \ln ppl^{t_2} \cdot 4 \approx 12.7 \end{aligned} \tag{3.4}$$

Therefore when using perplexity per token to compare models with different tokenizers, either the per-dataset negative log likelihoods must be used, or a common tokenizer must be agreed upon. E.g., if we agree on a tokenizer that would split the sentence into t_3 = [the, deforestation, <EOS>], then we get the same order of the two models as with phrase-level negative log likelihood.

$$\begin{aligned} ppl_1^w &\approx exp\frac{H_1}{3} = 134.4 \\ ppl_2^w &\approx exp\frac{H_2}{3} = 68.9 \end{aligned} \tag{3.5}$$

Perplexity-per-word can be computed either in the *micro-average* or the *macro-average* setting. In the *macro-average* setting, we accumulate the total unnormalized cross-entropy and the number of tokens and at the end we compute the perplexity $ppl = exp\frac{H}{\#\text{tokens}}$. In the *micro-average* setting, we compute the perplexity for each sample and then at the end we take the mean of all the perplexities.[2]

ConvAI2 Tokenization The perplexity per token on the ConvAI2 dataset [Dinan et al., 2020] is calculated by a published tokenizer in the macro-average setting. The ConvAI2 tokenizer operates on a word-level (see Section 1.2.1) and stores a vocabulary of common words. 0.49% of all words in the ConvAI2 dataset, which do not fit the tokenizer's vocabulary, are mapped to an *unknown* token. Words mapped to *unknown* token, as well as the *end-of-sequence* token are not considered when computing the perplexity.

3.1.2 Word-Level F1 Score

Word-level F1 score refers to a modification of a commonly used binary classification metric which is computed on the model-generated word sequences. First, we present the F1 score computation for a binary classifier, then we extend the definition to generated word sequences.

[2]We mention the *micro-average* setting as it is a common measure in frameworks such as HuggingFace, where the cumulative cross-entropy in training is reported as an average of per-batch cross-entropies.

The predictions of the binary classifier are divided into four categories (false and true positives, abbreviated to FP and TP respectively and false and true negatives, abbreviated to FN and TN) depicted in Table 3.1. First, *precision* (the proportion of correct predictions among all predictions) and *recall* (the proportion of correctly identified positive instances among all actual positive instances) are computed. The F1 score is then calculated to combine these metrics (Equation 3.6).

Prediction \ True Label	True	False
True	TP	FP
False	FN	TN

Table 3.1: Types of errors of a binary classifier.

$$\begin{aligned} \text{Precision} &= \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}} \\ \text{Recall} &= \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}} \\ \text{F1} &= \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \end{aligned} \tag{3.6}$$

The word-level F1 measure is based on the general F1 measure defined above, comparing a generated text (word sequence) to a reference human-written sequence (label). It uses specific definitions for true positive, false positive and false negative based on matching the reference, which we explain via the following examples:

Example 1 Let us assume that the word *"word"* occurs in both the reference and the prediction (see Table 3.2). Since it occurs three times in the reference, the first three occurrences in the prediction are counted as *true positives*, any other occurrences in the prediction are counted as a *false positive*.

Prediction:	"word word word word"
True Label:	"word word word"

Table 3.2: There are more occurrences of "word" in the prediction than in the reference.

Example 2 Now, let us look at the prediction in Table 3.3. The first three occurrences of word *"word"* in the prediction are counted as *true positive*. The one occurrence that is missing from the prediction is counted as a *false negative*.

Prediction:	"word word word"
True Label:	"word word word word"

Table 3.3: There are more occurrences of "word" in the reference than in the prediction.

3.1.3 Next Utterance Classification

Given a dialogue context, next utterance classification is a binary classification task to distinguish the positive candidate (the true next response in the dialogue from the dataset) from negative candidates (which may be randomly sampled from other dialogues in the dataset or some other source). For instance, each sample in the ConvAI2 dataset is accompanied by 19 negative candidate responses sampled from other dialogues in the dataset (see Section 2.3).

We assume that a dialogue model either directly produces a ranked list of all the candidates within the example, or assigns each candidate a score and we can sort the candidates by the score assigned. The performance of a dialogue model is measured by the hits@1 metric. If the positive example is ranked at the top of the list, then classifier receives one point, otherwise it receives zero points. The final result is the average over the entire dataset.

3.1.4 Other Employed Metrics

While the word-level F1 score (Section 3.1.2) gives some information about the dialogue model generation performance, we used several other simple metrics to further analyse the generations of the model. These focus more on the basic properties of the generated texts and do not need comparison to reference responses from the dataset.

Distinct-1, Distinct-2 The Distinct-1[3] and Distinct-2 metrics [Li et al., 2016b] compute the degree of diversity in generated outputs. They are defined as the fraction of the number of unique generated tokens (Distinct-1) or bigrams (Distinct-2) and the total number of words in the generated sentence (Equation 3.7).

$$\text{distinct-n} = \frac{\#\text{ distinct n-grams}}{\#\text{ total tokens}} \tag{3.7}$$

Persona-Copying The persona-copying metric is a ConvAI2 dataset [Dinan et al., 2020] specific metric that measures the average number of 4-grams per generation that are copied from sentences in the persona (see Section 2.2).

Fraction of Questions Fraction of questions is the proportion of the number of generated outputs containing a question mark to the total number of generated outputs.

Average Length in Words Average length in words indicates the average number of words in generated outputs as measured by the ConvAI2 tokenizer [Dinan et al., 2020] discussed in Section 3.1.2.

[3]The Distinct-1 metric measures the same quantity as the type-token ratio.

3.1.5 Alternative Metrics

In this section we list several other metrics that are used to evaluate open-domain dialogue models. For brevity, we present only an example for each whole class of similar metrics.

Word overlap metrics Word overlap metrics measure the co-occurence of characters, words or n-grams in the generation and the ground truth response, similarly to the word-level F1 metric discussed in Section 3.1.2. The BLEU score [Papineni et al., 2002] is a good example, focusing on n-gram precision. In Equation 3.8, the p_n refers to the n-gram precision, which is calculated similarly to the word-level precision discussed in Section 3.1.2. Since precision favors shorter sequences (when fewer tokens are produced, fewer false positive n-grams are produced) a separate term, the brevity penalty BP is introduced to penalize short sentences.

$$\begin{aligned} BP &= e^{\min(1-\frac{|\text{true response}|}{|\text{generation}|},0)} \\ BLEU &= BP \cdot \exp(\sum_{n=1}^{N} w_n \log p_n) \end{aligned} \tag{3.8}$$

Other word-overlap metrics used include ROUGE [Lin, 2004], METEOR [Banerjee and Lavie, 2005] or ChrF [Popović, 2015]. We do not use any of these metrics since they are generally stricter than word-level F1 and hence not particularly well suited for a task as open-ended as open-domain dialogue generation (most of these were designed for machine translation, where the range of possible responses is much lower) [Novikova et al., 2017].

Consistency Scores Consistency scores are based on natural language inference. Madotto et al. [2019] train a BERT model [Devlin et al., 2019] on the DNLI dataset [Welleck et al., 2019] (further discussed in Section 4.4.1), to predict whether the generation of the model directly follows (*entails*), contradicts or is neutral to any sentence in the persona description. Each contradiction contributes -1 to the metric, each entailment contributes +1 and neutral relationship does not contribute anything.

$$\begin{aligned} \mathbf{NLI}(g, p_i) &\in \{-1, 0, 1\} \\ \mathbf{C}(g) &= \sum_{i=1}^{|P|} \mathbf{NLI}(g, p_i) \end{aligned} \tag{3.9}$$

Models with higher scores generate utterances that follow the persona descriptions more and contradict them less. Madotto et al. [2019]'s metric is related to similar metrics for other tasks [Dziri et al., 2019, Dušek and Kasner, 2020]. We stick to more basic metrics in our own experiments due to their implementation simplicity and speed.

Classifier Metrics Another set of metrics uses classifiers, typically based on pretrained models. Mehri and Eskenazi [2020] finetune the RoBERTa model [Liu et al., 2019] on the next utterance classification task (discussed in Section 3.1.3) on the PersonaChat dataset [Zhang et al., 2018]. Afterwards the finetuned model

is used to estimate the probability of generated sequences. It is shown that such judgments correlate well with human annotators. Other examples of this class of metrics are ADEM [Lowe et al., 2016], or RUBER [Tao et al., 2018].

Besides a more complex implementation and higher computation costs, these metrics may not generalize well to different models or datasets [Sai et al., 2019].

Evaluation Based on Large Language Models The most recent trend is evaluating chat models using LLMs. Mendonça et al. [2023] use ChatGPT (discussed in Section 1.5.6) in an ensemble with XLM-RoBERTa [Conneau et al., 2020] to evaluate generations of models at the DSTC 11 competition [Rodríguez-Cantelar et al., 2023]. This ensemble provides evaluations which are best correlated with human evaluation across all the submissions to the competitions. However, the correlation still remains relatively low (0.4-0.5 range) and these metrics incur even higher computation costs than the previous types.

3.2 Human Evaluation

As discussed in Section 3.1.5, correlation of automatic metrics with human judgments on open-domain dialogue is still relatively low. On the other hand, human evaluation is the ultimate way of determining whether a given dialogue model is good. Human evaluation setups for dialogue are not standardized. The evaluation may consider the overall quality, or assess specific criteria such as readability, coherence, naturalness, or engagement [Santhanam and Shaikh, 2019], however, Howcroft et al. [2020] show that researchers attribute different meanings to these terms, and hence e.g., results in readability are not comparable across papers. We discuss the most general division into single-turn and multi-turn evaluation and into direct rating and relative ranking on a few examples.

Single-Turn Rating In the single-turn rating setting, the evaluator is presented with a dialogue context (the dialogue context may be complemented by additional information such as persona information for evaluation of systems on the ConvAI2 dataset [Dinan et al., 2020]), and a response to evaluate. For instance, the human evaluation of the P^2 Bot [Liu et al., 2020] is conducted in a setting where the evaluator is presented with the dialogue context, bot's persona and a generated response by a single model and assesses the quality of the generation on a Likert scale from 1 to 4.

Single-Turn Ranking In the single-turn ranking setting, the evaluator ranks responses from several models. In the ChatBot Arena [Chiang et al., 2024], an online evaluation tool, the evaluator asks a question and is presented with two generations by anonymised dialogue models. Their task is to evaluate which generation answers their question better and can choose from four possibilities: "A is better", "B is better", "Tie", "Both are bad". These pairwise evaluations are aggregated across model pairs and an ELO ranking is used to determine the best model. We use the single-turn ranking setting for evaluation of our models (see Section 5.7).

Multi-Turn Rating In the multi-turn rating setting, the evaluator may chat with the model for few turns and afterwards assess the quality of the whole dialogue. The multi-turn setting evaluates a dialogue model on its true designated task.

A multi-turn human evaluation was conducted in the ConvAI2 competition [Dinan et al., 2020] and the results decided the overall winner of the competition. The human annotator is paired with a model and instructed to "chat and get to know their partner." After a short conversation (up to 8 utterances) the evaluator is asked "How much did you enjoy talking to this user?" on a scale of 1-4.

4. Related Work

In this chapter we discuss selected models previously reported on the ConvAI2 dataset. In Section 4.1 we review the results of automatic metrics reported on the ConvAI2 dataset. We follow with a discussion of selected models which inspired us for our experiments conducted in Chapter 5. We experiment with multi-task finetuning conducted by [Wolf et al., 2019] that we discuss in Section 4.2. The decoding scheme used in this approach inspired us to train a separate ranking model. Liu et al. [2020] use reinforcement learning to improve their model (Section 4.3), which inspired us to try the direct preference optimization finetuning. Lastly, we discuss the LMEDR model, which reaches the state-of-the-art performance in all the automatic metrics on the ConvAI2 dataset [Dinan et al., 2020]. We directly compare to LMEDR in our experiments, and use the model's outputs as basis for further reranking.

4.1 Results of Automatic Metrics

In Table 4.1 we see the reported performance of various models on the ConvAI2 dataset. We only briefly describe BART [Lewis et al., 2020] and DialogueDodeca [Shuster et al., 2020] below, since except for pretraining on a different dataset, they do not introduce any specific dialogue modeling related novelty. The rest of the models is described in detail in the following sections.

BART Lewis et al. [2020] introduce the transformer encoder-decoder (see Section 1.2.7) BART model and reach the contemporary state-of-the-art perplexity and F1 score by finetuning with language modeling objective.

DialogueDodeca Shuster et al. [2020] train a transformer encoder-decoder model (see Section 1.2.7) on 12 different dialogue modeling datasets including the ConvAI2 dataset.

Model	#Params	F1 score	PPL	hits@1
TransferTransfo [Wolf et al., 2019]	124M	19.09%	17.51	82.1%
P^2 Bot [Liu et al., 2020]	124M	19.77%	15.12	81.9%
BART [Lewis et al., 2020]	406M	20.72%	11.85	-
DialogueDodeca [Shuster et al., 2020]	87M	21.3%	-	-
LMEDR [Chen et al., 2023]	406M	21.99%	10.99	89.5%

Table 4.1: Performance of models on automatic evaluation on the ConvAI2 dataset [Dinan et al., 2020].

4.2 TransferTransfo

The TransferTransfo approach [Wolf et al., 2019] is the winner of the automatic metrics part of the ConvAI2 competition [Dinan et al., 2020]. Wolf et al. [2019]

finetune the pretrained GPT model [Radford and Narasimhan, 2018] (see Sections 1.4.1-1.4.2).

In addition to standard token and position embeddings a separate set of *dialogue state embeddings* is trained so that the model could learn to distinguish parts of the input corresponding to the persona description, the past dialogue context and the response (Figure 4.1).

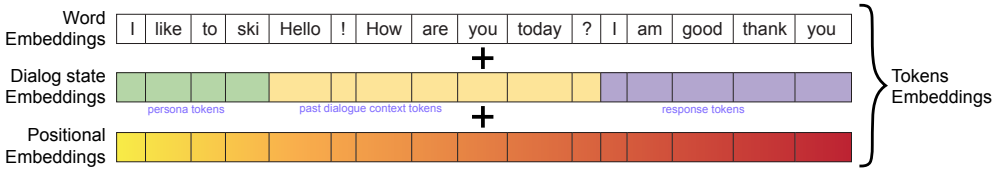


Figure 4.1: The final embedding that is processed by the GPT backbone model is a sum of three separate embeddings. Dialog state embeddings are randomly initialized at the start of finetuning. Figure taken from Wolf et al. [2019].

The finetuning is conducted in a multi-task setup, where the loss is a weighted sum of multiple components. In Section 2.3 we mentioned that for each utterance a set of 19 negative candidates is randomly sampled for the next utterance classification task, where the model should distinguish between the true response and negative candidates. Wolf et al. [2019] add a linear layer with a sigmoid activation on top of the transformer decoder's representation of the `<eos>` token (Figure 4.2) and jointly train the model for the classification task.

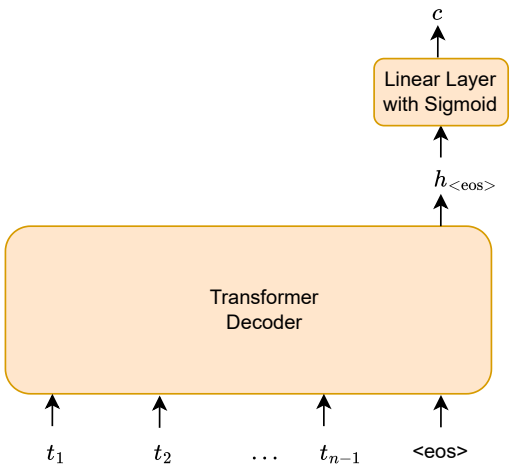


Figure 4.2: In order to use a transformer decoder for classification, Wolf et al. [2019] add a linear layer with sigmoid activation and train it on the next utterance classification task.

The language modeling loss is computed only for positive candidates and the next utterance classification is modeled with binary cross-entropy (Equation 4.1). The best results are obtained by setting $\lambda = 2$, i.e., making the language modeling loss two times more important.

$$
\begin{aligned}
\text{is\_positive} &\in \{0, 1\} \\
\mathcal{L}_{CLS} &= BCE(c, \text{is\_positive}) \\
\mathcal{L} &= \mathcal{L}_{CLS} + \lambda \cdot \text{is\_positive} \cdot \mathcal{L}_{LM}
\end{aligned} \tag{4.1}
$$

4.2.1 Decoding Scheme

For decoding, Wolf et al. [2019] use the *generate-and-rank* approach [Challa et al., 2019]. In this method, one model (or model component) is used to generate candidates and another model (or model component) is used to rank the candidates. The best ranked response is selected as the response of the system.

Wolf et al. [2019] generate candidate utterances using beam search with sampling with a beam size of 4 (Section 1.2.4). The candidates are ranked by a weighted sum of the length-normalized negative log likelihood $\mathcal{S}_{\text{LM}}$ assigned by the language model and the score $\mathcal{S}_{\text{CLS}}$ assigned by the classification head.

$$
\mathcal{S} = \mathcal{S}_{\text{LM}} + \lambda \cdot \mathcal{S}_{\text{CLS}} \tag{4.2}
$$

Wolf et al. [2019] note that when λ is higher the utterance with the highest score "sticks more closely to the provided personality," but it "reduces the diversity of the dialogue."

4.3 P^2 Bot

Similarly to the TransferTransfo model, the Persona Perception Bot (P^2 Bot) [Liu et al., 2020] is a finetuned version of the GPT model [Radford and Narasimhan, 2018]. Contrary to TransferTransfo discussed in above, the P^2 Bot makes use of reinforcement learning.

The finetuning process is conducted in a two-stage manner. In the first stage, the GPT model is finetuned on the ConvAI2 dataset [Dinan et al., 2020] in the same multi-task setup as TransferTransfo (Section 4.2).

In the second stage, two copies of the model, $\mathcal{A}_{\text{frozen}}$ and $\mathcal{A}_{\text{FT}}$ communicate in a dialogue setting and the $\mathcal{A}_{\text{FT}}$ is finetuned through reinforcement learning.[1]

While the used reinforcement learning algorithm (REINFORCE, [Williams, 1992]) and its application to the NLG are standard, the contribution of Liu et al. [2020] lies in the so-called *reward shaping*, i.e., specifying the reward for the agent $\mathcal{A}_{\text{FT}}$ after a dialogue, which specifies its weights update.

Liu et al. [2020] design a reward that is a sum of three separate terms. The *language modeling loss* estimated by the $\mathcal{A}_{\text{frozen}}$ model rewards producing a human-like response. The *discourse coherence loss* estimated by the classification head of $\mathcal{A}_{\text{frozen}}$ rewards "establishing links in meaning with context." Finally the *Mutual Persona Perception* score, is a score evaluated by a separate BERT model [Devlin et al., 2019] finetuned to discriminate utterances produced by a specific persona.

[1] For a detailed introduction into reinforcement learning see Reinforcement Learning: An Introduction [Sutton and Barto, 2018].

4.4 LMEDR

According to our knowledge the LMEDR model [Chen et al., 2023] is the best scoring model on all automated metrics commonly measured on the ConvAI2 dataset [Dinan et al., 2020]. Chen et al. [2023] use a two-stage finetuning of the BART-Large model [Lewis et al., 2020].

In the first stage, the model is trained on a modification of a dialogue natural language inference task (see Sections 4.4.1), in the second stage the model is trained with a multi-task setup similar to TransferTransfo discussed in previous section (see Section 4.4.2). The training is cyclical, where the first and second stage are iterated until convergence.

Chen et al. [2023] use task-specific architecture augmentations, dubbed *Entailment Relation Memory* and *Dialogue Discourse Memory*. Architecturally, each memory is a set of learned vectors together with a linear attention layer over the learned vectors (see Section 1.2.6 and Figure 4.3). The *Entailment Relation Memory* is hypothesized to learn "to store entailment relations for persona consistency," i.e., thanks to this module, the model should generate utterances which are less contradictory with respect to the persona description. The *Dialogue Discourse Memory* is a component that is hypothesized to "ensure discourse coherence," i.e., thanks to the module, the model should generate utterances which are less contradictory with respect to the past dialogue context.

4.4.1 LMEDR: First Stage of Training

During the first stage, Chen et al. [2023] train on the DialogueNLI (Dialogue Natural Language Inference) dataset [Welleck et al., 2019].

NLI NLI [Williams et al., 2018] is a task where the model should predict whether a provided pair of texts, commonly called a *premise* and a *hypothesis* is in the relation of *entailment* (*hypothesis* logically follows from *premise*), *contradiction* (*hypothesis* contradicts *premise*), or *neutrality* (*premise* and *hypothesis* are not in any relation).

Dialogue NLI In Dialogue NLI [Welleck et al., 2019], all persona description sentences from the PersonaChat dataset [Zhang et al., 2018] are used as premises. A hypothesis is either another sentence from persona descriptions or an utterance from a dialogue in the dataset. By default, all sentences are assumed to be in a neutral relation. Welleck et al. [2019] crowdsource annotations for dialogue utterances and persona description sentences in form of triples *(<category>, <relation>, <category>)* e.g. *(i, have_pet, dog)*. A hypothesis *entails* a premise if they are annotated with the same triple.

It is much harder to annotate the *contradiction*. Barring the obvious cases of logical contradictions such as sentences *"I have a dog."* and *"I do not have a dog."* Welleck et al. [2019] note that one should also consider a pair of sentences "not likely to be said by the same persona." to be a contradiction. For example pairs (*"I'm looking forward to going to the basketball game this weekend!"*, *"I don't like attending sporting events."*) and (*"I'm a lawyer."*, *"I'm a doctor."*) are also annotated as contradictions. A set of heuristics is designed either to find such

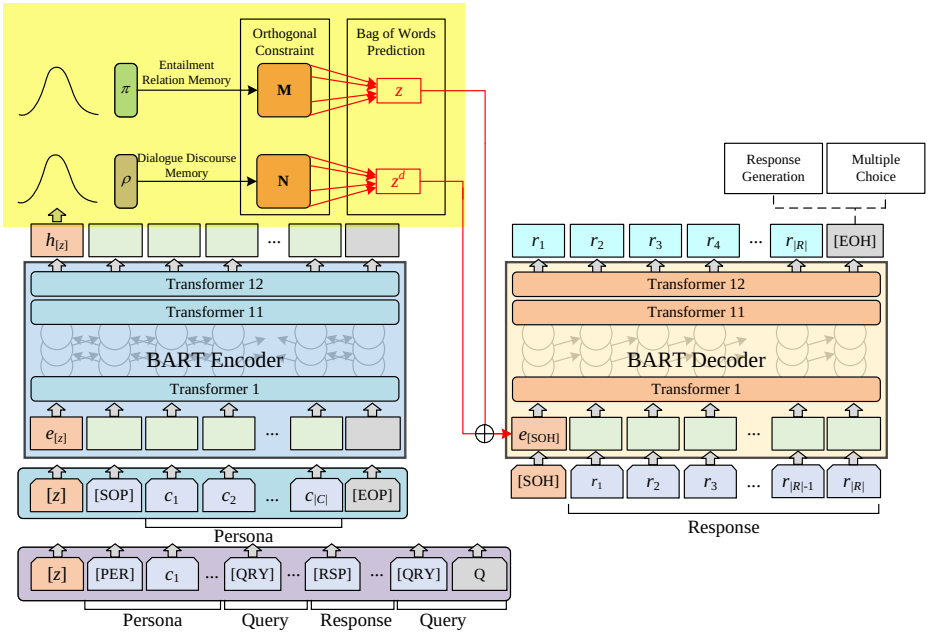


Figure 4.3: Diagram of the LMEDR model. The highlighted area displays the task-specific architectural changes. Figure taken from Chen et al. [2023].

pairs in the dataset, or to generate them. The generation is done by replacing numerical information ("I am 40 years old." → "I am 50 years old").

LMEDR: First Stage of Training Chen et al. [2023] finetune the BART model on pairs with entailment relation for conditional generation of hypotheses conditioned on premises.

4.4.2 LMEDR: Second Stage of Training

In the second stage, the ConvAI2 dataset[Dinan et al., 2020] is used for finetuning. The *Entailment Relation Memory* is frozen (i.e., no weight updates on this module are done).

Chen et al. [2023] follow the multi-task setup from TransferTransfo (discussed in Section 4.2) and use several additional tasks, e.g., bag of words loss [Zhao et al., 2017] where a classifier is trained to predict which words should be in the generation conditioned on the latent variable from *Dialogue Discourse Memory* and correlation loss between *Dialogue Discourse Memory* and *Persona Entailment Memory* (Figure 4.3).

During inference, a standard beam search with beam size 2 is used, i.e., unlike TransferTransfo (Section 4.2), only the language model score is used and the classification head is not involved in the ranking.

5. Experiments

In Chapter 4, we explored different strategies for training small language models on the ConvAI2 dataset. In this chapter, we build on these findings and conduct our own experiments on this data, using both finetuning (for smaller models) and prompting for LLMs.

In Sections 5.1 and 5.2, we establish baseline performance and improve it with more efficient decoding methods. Afterward, in Sections 5.3 and 5.4, we explore a multi-task learning setup similar to TransferTransfo (discussed in Section 4.2) and build a two-stage generation pipeline. Then we explore direct preference optimization (Section 5.5), a method similar to the reinforcement learning setup of P^2 Bot (see Section 4.3). We conclude our experiments by prompt engineering GPT-3.5 (Section 5.6). To complement the evaluation by automatic metrics, we choose the prompt-engineered GPT-3.5 and our best models according to the F1 score for a round of human evaluation, where three annotators evaluate generations by three selected models (Section 5.7). In Section 5.8 we discuss the results of all the experiments in this chapter.

5.1 Baseline

We are interested in the effects of individual modifications of training and inference pipeline to the downstream performance of the model on our task. Therefore, we start with a very simple baseline and come up with modifications inspired by the literature. First, we discuss the selection of the GPT-2 family of models for the majority of experiments in this chapter (Section 5.1.1) and the model input format that we use throughout the experiments (Section 5.1.2). We follow by an assessment of performance on different automatic metrics, namely the model-internal perplexity on the validation data (Section 5.1.3), hits@1 on next utterance classification (Section 5.1.4) and generation performance in terms of F1 reference match and other text properties (Section 5.1.5). Lastly, we further analyse the generated responses (Sections 5.1.6, 5.1.7), and discuss the selection of the model for further experiments (Section 5.1.8).

5.1.1 Baseline Models

To establish a baseline, we use the pretrained checkpoints of the following models: GPT-2-small,[1] GPT-2-medium and GPT-2-large [Radford et al., 2019]. The models are trained to predict the next utterance conditioned on the input. We use the standard conditional language modeling loss (see Section 1.2.5).

Due to available resources (GPUs with 12GBs of memory) the biggest model that is feasible to train without any quantization is GPT-2-medium (about 355M parameters). GPT-2-large (about 762M parameters) is the biggest model that we could fit in the GPU memory with 8-bit optimizer quantization [Dettmers et al., 2022b] and gradient checkpointing [Chen et al., 2016]. While it may be feasible to train bigger models using LoRA [Hu et al., 2022] and QLoRA [Dettmers

[1] GPT-2 model according to naming on HuggingFace (`https://huggingface.co/openai-community/GPT-2`)

et al., 2023] approaches, we decided to not experiment with these approaches due to shortage of time. While we could have placed more emphasis on comparing finetuning performance of a bigger range of models, we are more interested in the effects of different finetuning approaches than effects of using different pretrained models.

We expect bigger models to produce more dialogue responses that score higher in automatic evaluation, as increasing the capacity of the language model "improves performance in a log-linear fashion accross tasks." Radford et al. [2019]

We optimized the final hyper-parameters for training based on analysis of validation negative log likelihood over several experiments. We found that using the Adam optimizer [Kingma and Ba, 2015], learning rate of $1.5 \cdot 10^{-6}$ and a batch size 16 yields good enough results for all the model sizes that we experiment with.

5.1.2 Model Input Format

Similarly to Liu et al. [2020], we add three new tokens to the model's vocabulary. These tokens are put at the beginning of different sections of model's input. Each sentence in the persona description is prepended with the <persona> token. Each sentence in the dialogue is prepended either with the <bot> token (if it's produced by the bot's persona) or with the <partner> token. Figure 5.1 shows an example input of the model.

<persona>i read twenty books a year. <persona>i'm a stunt double as my second job. <persona>i only eat kosher. <persona>i was raised in a single parent household. <partner>hello what are doing today? <bot>i am good, i just got off work and tired, i have two jobs. <partner>i just got done watching a horror movie <bot>**i rather read, i've read about 20 books this year.<|endoftext|>**

Figure 5.1: Example input of the model. The language modeling loss is calculated only on the bold text. The shading is added only for better readability.

5.1.3 Evaluation of Perplexity

As expected, larger models reach lower perplexities (Table 5.1). We note that the performance of GPT-2-small roughly matches the reported performance of the TransferTransfo model (see Section 4.2); however, it is more than two points worse than the performance reported by P^2 Bot (see Section 4.3). The perplexity measured on GPT-2-medium beats the perplexity of the P^2 Bot; however, the used model has three times more parameters (124M vs 355M).

Neither of our models matches the reported perplexities of BART and LMEDR models. However, upon inspection of the code of the LMEDR model,[2] we found that in fact they compute perplexity per token with the BART tokenizer and that they include the predicted probability of the end of sequence

[2]Chen et al. [2023] released their code and model at `https://github.com/Chenrj233/LMEDR`

Model	PPL-T	PPL-W
TransferTransfo	-	17.51*
P^2 BOT	-	15.12*
BART	-	11.95*
LMEDR	-	10.99*
GPT-2-small	15.82	17.53
GPT-2-medium	12.94	14.32
GPT-2-large	12.41	13.71

Table 5.1: Perplexity per word using the ConvAI2 tokenizer (PPL-W, Section 3.1.1) and perplexity per token of the GPT-2 tokenizer (PPL-T) [Radford et al., 2019], for our baseline methods and models discussed in Chapter 4. "*" = as reported by the respective paper.

token in the computation of the final perplexity (we show drawbacks of comparing per-token perplexities for models with different tokenizers in Section 3.1.1). Using LMEDR's public model checkpoint, we remeasured the perplexity of the LMEDR model using our code (Table 5.2) and found that in terms of perplexity, the GPT-2-medium model of the same size (355M vs 406M) slightly beats the performance of the LMEDR model.

As Lewis et al. [2020] do not publish the finetuned checkpoints of BART, we can only hypothesize about the used tokenization method when measuring the perplexity of the BART model. While they claim to have used the ConvAI2 tokenizer for the measurement, in our informal experiments with the BART model and the ConvAI2 tokenizer, the perplexity per word was roughly on par with GPT-2-medium.

Method	PPL
Reported Perplexity	10.99
BART Tokenizer + </s>	10.95
BART Tokenizer	12.59
ConvAI2 Tokenizer	15.59

Table 5.2: Perplexity per token of the LMEDR model [Chen et al., 2023] as measured by different tokenizers.

5.1.4 Evaluation of Next Utterance Classification

The next utterance classification task (described in Section 3.1.3) consists of discriminating 19 negative candidates from the true response in the dataset.

Baseline models are not explicitly trained for the next-utterance classification. We rank all candidates by the length-normalized negative log likelihood (Equation 5.1) and the candidate with the highest score from 20 candidates is taken as the prediction of gold utterance.

$$\mathcal{S} = \frac{1}{|T|} \sum_{t=1}^{T} \log \hat{p}(x_t) \tag{5.1}$$

In Table 5.3 we see that this method is inferior to using a ranking head

explicitly trained for the next utterance classification task that is used in the related work.

Model	Hits@1
TransferTransfo	82.1%
P^2 Bot	81.9%
LMEDR	89.5%
GPT-2-small	24.53%
GPT-2-medium	29.56%
GPT-2-large	31.57%

Table 5.3: Performance of baseline models on hits@1 on next utterance classification.

5.1.5 Evaluation of Generations

The F1 score is the only automatic metric commonly measured on the ConvAI2 dataset [Dinan et al., 2020] that measures the generation performance of models (see Section 3.1). In order to better explain the F1 score, we include additional metrics, namely persona-copying, fraction of questions, distinct-2 and average length in words (discussed in Section 3.1.4). We generate with greedy decoding (Section 1.2.4). In Section 5.2 we use more advanced decoding methods with the best checkpoint of the GPT-2-medium model.

In Table 5.4 we see that the finetuned GPT-2-small scores 0.3 points under the performance of TransferTransfo, while the performance of GPT-2-medium is similar to the performance of P^2 Bot. We note that the results of GPT-2-small and GPT-2-medium are measured on greedy-decoded generations, while all the methods from related work (see Chapter 4) make use of more advanced decoding methods.

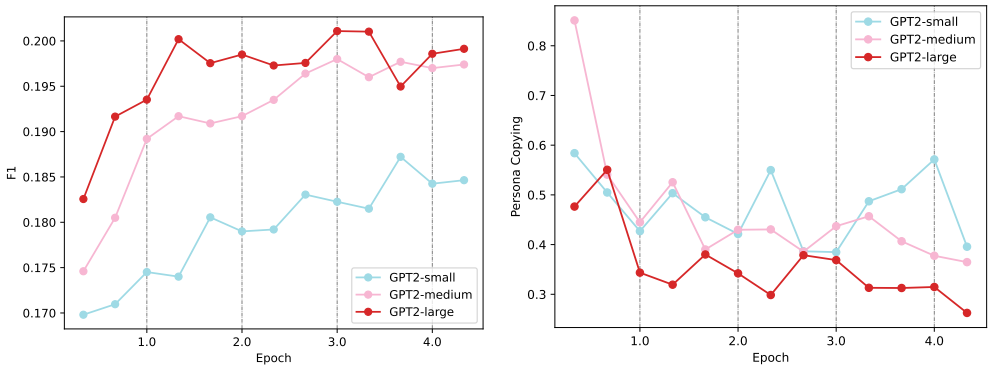
Method	F1 score
TransferTransfo	19.09%
P^2 Bot	19.77%
LMEDR	21.99%
GPT-2-small	18.72%
GPT-2-medium	19.80%
GPT-2-large	20.11%

Table 5.4: Performance of baseline models and models from related work on F1 score.

Over the course of the training, the F1 score plateaus at 18.5 for GPT-2-small, 19.7 for GPT-2-medium and 19.9 for GPT-2-large (see Figure 5.2a).

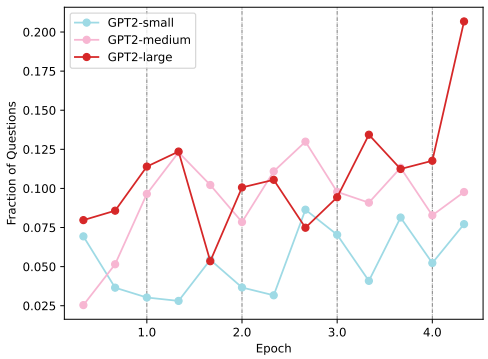
At the beginning of the training, models "cheat" by copying long chunks of text from the persona. As the training progresses, models copy less from the persona (Figure 5.2b), learn to ask more questions (Figure 5.2c), use more diverse language (Figure 5.2d) and produce slightly longer utterances (Figure 5.2e).

Larger models use a more varied vocabulary (Figure 5.2d) and copy fewer facts from the persona (Figure 5.2b).

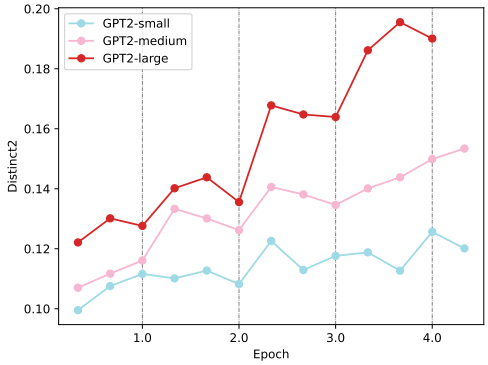


(a) F1

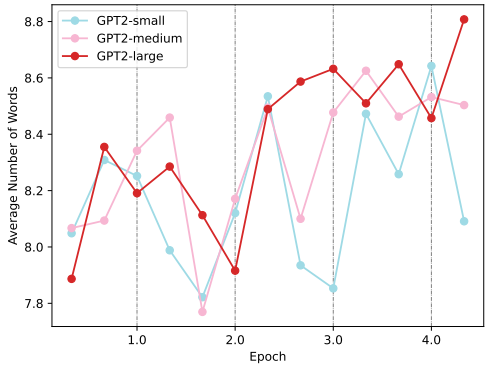
(b) Persona-copying



(c) Fraction of questions



(d) Distinct-2



(e) Average number of words

Figure 5.2: Evolution of different metrics measured on the validation set over the course of training of the GPT-2-family models with handpicked hyper-parameters.

5.1.6 Manual Analysis of Generations

We picked 10 random samples and evaluated the quality of generated outputs on these 10 samples by each model in all the training epochs in detail. In general, models learn to include facts from personas in their responses. However, it seems that models mention these facts excessively, i.e., almost all the responses generated by models contain a fact from the persona. We prefer responses from larger models as they are less repetitive and they react better to the dialogue partner.

Problems with Structure of the Input In order to be successful in the task, the model needs to understand the structure of the input. As explained in

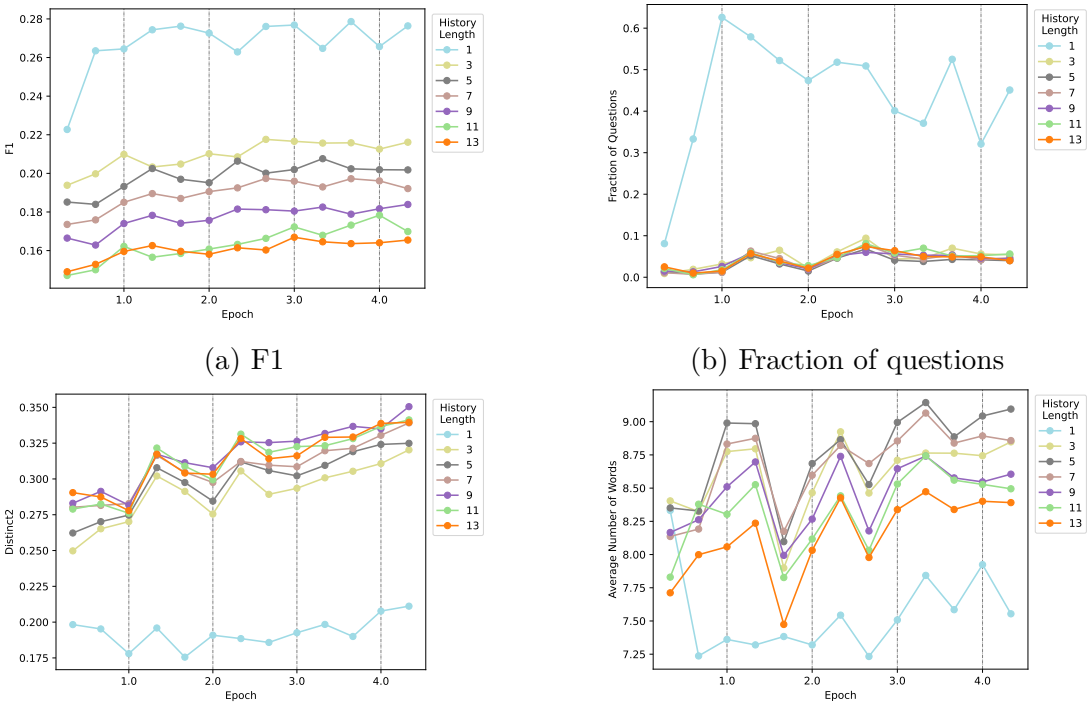
Section 5.1, the parts of the input have different meanings and are distinguished by special prepended tokens (<persona>, <bot>, or <partner>).

It seems that the models are prone to confuse an utterance from the dialogue history in the form *"I like <activity>"* to a persona fact. E.g., after a dialogue partner says that they like something, the bot starts responding that it likes the mentioned thing too.

5.1.7 Generations by Length of Dialogue History

During manual analysis (Section 5.1.6) we observed that the model responses are of worse quality when the dialogue history is longer. In order to verify this observation we split generations according to the length of the dialogue history and we measure metrics for each length separately (Figure 5.3).

Firstly, there is a visible step between the measured quality of the first response of GPT-2-medium (length of dialogue history 1) and the remaining responses. Usually the first and second utterances in the dialogue follow the same pattern. Firstly the partner asks *"Hi, how are you?"* and then the bot responds *"I'm fine, thank you. I do <activity> as a hobby, what about you ?"* This observation is supported by the reduced vocabulary of first responses (Figure 5.3c) and increased fraction of questions (Figure 5.3b).



(a) F1

(b) Fraction of questions

(c) Distinct-2

(d) Average number of words

Figure 5.3: Metrics measured on outputs generated by GPT-2-medium which are split by the history length.

In Figure 5.3a, we see that with progressing dialogue length, the F1 score gets smaller; however, we cannot explain this phenomenon with the other measured generation metrics. Intuitively we attribute the decrease to lower predictability of

later utterances. After a relatively repetitive conventional politeness expressions in the first few turns, the remainder of the dialogue is very open.

5.1.8 Conclusion

We showed that a baseline without extensive hyper-parameter tuning initialized from GPT-2 family of models matches or exceeds the performance of models initialized from the older GPT model, i.e., we confirm the importance of pretraining. We analyse the quality of generations by the baseline model and discuss what the models learn over the course of training and how they react to different lengths of dialogue history.

In the following sections, we use only the GPT-2-medium model, since in our conditions it offers the best tradeoff between speed of training and performance in the inference among the models from the GPT-2 family [Radford et al., 2019].

5.2 Advanced Decoding Methods

Up until now, we only used the greedy decoding method for generating utterances with trained models. In Section 1.2.4 we discussed several other generation methods that approximate more closely the maximum likelihood objective in Equation 5.2. In this section we use these generation methods with the GPT-2-medium checkpoint that scored highest in the F1 score (discussed in Section 5.1).

$$\hat{y}_{1:\hat{T}} = \arg\max_{y_{1:T}} p(y_{1:T}|\text{persona}, x_{1:N}) \tag{5.2}$$

In the following sections our goal is to find a method that results in the highest validation F1 score. For achieving this goal we explore the beam search decoding with several beam sizes (Section 5.2.1) and experiment with other decoding methods such as diverse beam search, or beam search with sampling (Section 5.2.2).

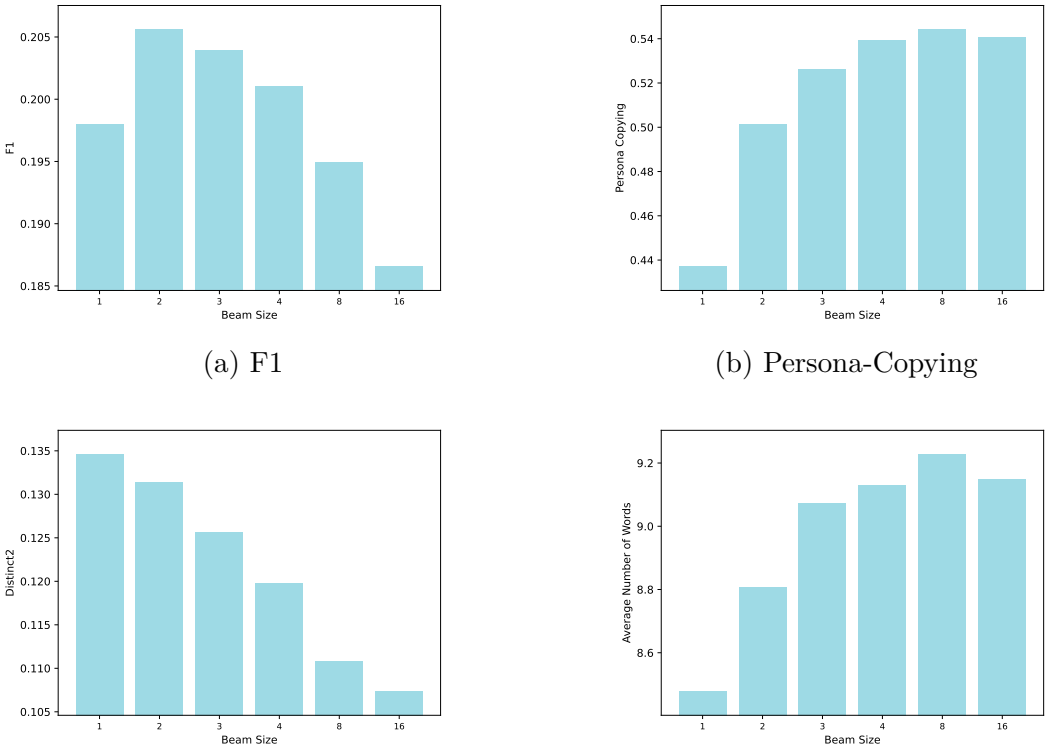
5.2.1 Beam Search

We explore the performance of the GPT-2-medium model with 5 different beam sizes $\{1, 2, 3, 4, 8, 16\}$. In Figure 5.4a, we see that the best F1 score (20.57%) is reached with beam size two and that only beam sizes lower than or equal to four outperform the greedy decoding (greedy decoding corresponds to beam search with beam size 1). Yang et al. [2018] call this phenomenon *a beam search curse*. While the larger beam size surely finds a result with higher or equal likelihood, the measured performance of word-overlap metrics deteriorates.

With larger beam sizes, the generations are longer (Figure 5.4d), contain more 4-grams from personas (Figure 5.4b) and their vocabulary is less rich (Figure 5.4c).

5.2.2 Beam Search Modifications

In this section we explore two other decoding methods, diverse beam search [Vijayakumar et al., 2018] and beam search with sampling [Wolf et al., 2019] dis-



(a) F1

(b) Persona-Copying

(c) Distinct-2

(d) Average number of words

Figure 5.4: Beam Search Results measured on outputs generated by the best performing checkpoint of the GPT-2-medium model under greedy decoding.

cussed in Section 1.2.4.

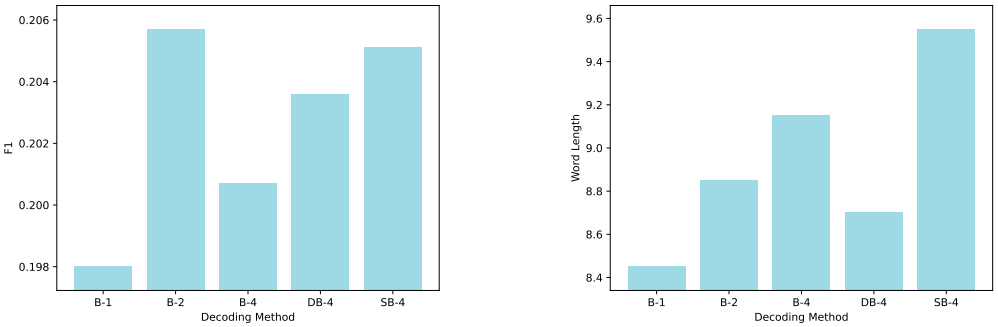
The intuition behind both the diverse beam search and the beam search with sampling is that by exploring more diverse generations, a better (according to maximum likelihood) generation may be found. Indeed that is the case as both beam search modifications with beam size 4 find solutions that achieve better F1 score than the basic beam search with the same beam size.

5.2.3 F1 Score Potential

We hypothesise that better F1 scores can be reached with a two-stage system. In the first stage a standard decoding algorithm (such as beam search, or modifications of beam search presented in Section 5.2.2) is used for generating candidate outputs. In the second stage a ranking model ranks the candidate outputs and the highest ranked candidate is selected as the output of a system.

Decoding Method	F1 score			
	Max	Min	NLL-Selected	Random
Beam Search	24.96%	16.03%	20.07%	20.37%
Diverse Beam Search	**27.94%**	**12.27%**	**20.36%**	**19.63%**
Beam Search with Sampling	24.79%	16.49%	20.46%	20.48%

Table 5.5: Comparison of different rankings of candidates generated by beam search modifications with beam size 4.



(a) F1 score

(b) Length-in-Words

Figure 5.5: Metrics measured on candidates generated by the GPT-2-medium model checkpoint that performs the best according to F1 score under greedy decoding, discussed in Section 5.1
Note: B = Beam Search, DB = Diverse Beam Search, SB = Beam Search with Sampling
Note: Number on the right of the hyphen is the beam size

To verify this hypothesis we store four candidates for each validation example, disregarding the default ordering given by length-normalized negative log likelihood. We then rerank the samples by F1 score (either min or max), or randomly. We observe three interesting phenomena:

1. The default length-normalized negative log likelihood ordering is far from optimal with respect to F1 (about 5-7 points below the upper bound for beam size 4). More interestingly, *random sampling outperforms the length-normalized negative log likelihood ordering for beam search and beam search with sampling* (Table 5.5).

2. The candidates produced by diverse beam search are the most differing, since the range between oracle minimum and oracle maximum performance is the largest. Manual analysis of 10 randomly selected samples confirms this finding. While candidates produced by beam search and beam search with sampling have the same syntactical structure and meaning and differ only by few words, the candidates produced by diverse beam search differ in meaning.

3. A ranking model can reach significantly higher F1 score than any method presented in the literature (the SoTA is 21.98% by the LMEDR model [Chen et al., 2023] discussed in Section 4.4).

5.3 Next Utterance Classification

Following the TransferTransfo [Wolf et al., 2019] (Section 4.2), we add a sigmoid layer that processes the language model's representation of the `<eos>` token to produce the ranking score. In the following text we call this layer *the ranking head* (Figure 5.6). We train a model with a ranking head and a language modeling head jointly with language modeling and next utterance classification objectives. We investigate three hypotheses.

Firstly, models that train the ranking head for the next utterance classification objective (see Section 5.3.1) reach about three times better score than our baselines on the hits@1 metric for next utterance classification (Table 5.3). We

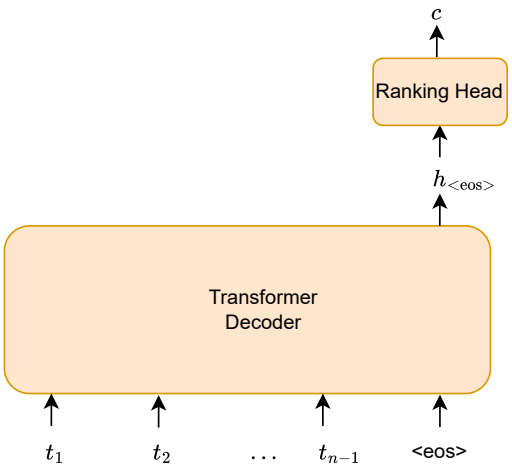


Figure 5.6: Schema of a transformer decoder model with a ranking head.

hypothesise that all of this improvement can be attributed to the joint training of the language model and a ranking head.

Next, the multi-task learning could help to optimize the other metrics as well.

Lastly, we use the generate-and-rank approach, where the utterances generated by the model are subsequently ranked by the ranking head. Following the estimation of the potential of such a system that we made in Section 5.2.3, we hypothesise that it may reach higher F1 score than the generation model alone.

5.3.1 Multi-Task Learning

The dataset contains 19 randomly sampled negative candidates for each positive gold utterance (Section 2.3).

To prevent overt imbalance in training data, the model is randomly fed with positive and negative utterances. In our experiments, we used a setting with 25% positive and 75% negative utterances, i.e., using only three randomly selected negative candidates per each positive candidate. When the model is fed with a positive input, we train both the language modeling head and the ranking head by summing the loss functions for bot tasks similar to Wolf et al. [2019] (see Section 4.2) (Equation 5.3). Otherwise we train only the ranking head.

$$\mathcal{L} = \mathcal{L}_{\text{cls}} \cdot \lambda_{\text{cls}} + \mathcal{L}_{\text{LM}} \tag{5.3}$$

The value of λ_{cls} specifies how much weight to put on the classification loss. Since the dataset contains more negative samples than positive, Wolf et al. [2019] use values between $(0.0, 1.0]$, after some tuning we found out that value $\lambda_{\text{cls}} = 0.5$ gives the best results for all the measured automatic metrics.

Next-Utterance Classification Results Our hypothesis of joint training improvement is confirmed by the results, as the hits@1 score of the jointly-trained model exceeds the score of TransferTransfo [Wolf et al., 2019] and P^2 bot [Liu

et al., 2020] and is close to the performance of the state of the art LMEDR [Chen et al., 2023] (Table 5.6).

Model	Hits@1
TransferTransfo	82.1%
P^{2}BOT	81.9%
LMEDR	89.5%
GPT-2-medium (LM-only)	30.41%
GPT-2-medium (ranking head)	89.32%

Table 5.6: Hits@1 on next utterance classification task.

Response Generation Results On the other hand, the model trained in the multi-task setup either matches or slightly under-performs the GPT-2-medium model trained only with the language modeling objective. Hence, it seems that the multi-task training does not increase the performance of the model on the response generation task, according to automatic metrics.

Model	perplexity	F1 Score
TransferTransfo	17.51	19.09%
P^2 BOT	15.12	19.77%
LMEDR	10.99	21.99%
GPT-2-medium (LM-only)	14.32	19.8% (20.57%)
GPT-2-medium (ranking head)	14.85	19.8% (20.43%)

Table 5.7: Perplexity and F1 score measured on our models compared to reference models from the literature. Numbers in brackets refer to F1 score under beam search decoding with beam size two.

5.3.2 Two-Stage Generation

We use the GPT-2-medium model with ranking head to generate four candidate utterances for each sample with the diverse beam search decoding and we evaluate several methods of selecting of the resulting utterance.

The ranking head trained on the next utterance classification objective is not a good predictor of F1 score as the two-stage system with ranking head as a ranker scores lower than the selection according to length normalized negative log likelihood (Table 5.8).

Following TransferTransfo [Wolf et al., 2019] we also use a weighted combination of the score from the ranking head $\mathcal{S}_{\text{rank}}$ and the length-normalized negative log likelihood $\mathcal{S}_{\text{LM}}$ (Equation 5.4).

$$\mathcal{S} = \lambda_C \cdot \mathcal{S}_{\text{rank}} + \lambda_{\text{LM}} \cdot \mathcal{S}_{\text{LM}} \tag{5.4}$$

The improvement found using an extensive grid search over values of λ_C and λ_{LM} is only marginal (see Table 5.8). The ranking head trained on negative candidates drawn from other dialogues does not prove to be a good ranker in a two-stage setup. We hypothesise that the reason is the difference between training

Method	F1-Score
Random (averaged over 20 runs)	19.58%
Ranking Head Selection	19.87%
Length-Normalized NLL	20.24%
Weighted Combination	20.30%
Oracle Max F1	27.97%

Table 5.8: Comparison of different ranking methods for a setup where for each sample, four candidate utterances are generated by the GPT-2-medium model with ranking head and a ranking method is used to rank the utterances.

and validation tasks. In training, the classification head learns to discriminate one positive candidate from 19 candidates sampled from other dialogues. Therefore, we think that it learns to look for inconsistencies and contradictions between a candidate and either the past dialogue context, or the persona description. In the two-stage setup, the classification head is used to rank utterances which are produced by a model which in itself is trained (in a semi-supervised way) to not generate inconsistencies and contradictions (see Section 2.2.1).

These utterances differ much less than the set of 19 negative candidates and a true positive utterance and all of them either react to the past dialogue content, or present some fact from the persona description. Therefore instead of training the ranking head with the next utterance classification objective, a ranker of candidates generated by a single model may be able to improve the overall F1 score.

5.4 Learning to Rank

In Section 5.3, we hypothesized that a generate-and-rank approach may improve the performance of a dialogue model in terms of F1 score. However, the GPT-2-medium model with a jointly trained ranking head and language modeling head does not prove so (Section 5.3.2).

In this section, we conduct another round of experiments with the generate-and-rank approach. However, this time, we only focus on developing a better model for the second stage. We train a separate GPT-2 model with a ranking head (Figure 5.6) to predict the ranking of utterances according to the F1 score, contrary to the joint training setup described in Section 5.3.2. The preliminary experiments are conducted with the GPT-2-small model to speed up the development, and the final experiment is evaluated with the GPT-2-medium model.

We only train the model for ranking using the learning-to-rank approaches. Namely, we train in a pointwise way [Caruana et al., 1995], where the ranking model is trained to estimate the F1 score of the candidate response, and in a pairwise way. We experiment with two approaches to pairwise training, RankNet [Burges et al., 2005] and LambdaRank [Burges et al., 2006]. In both these methods, the ranking model is trained to select a response with a higher F1 score from a pair of responses. While RankNet uses vanilla classification loss, in LambdaRank, the classification loss is weighted by the absolute value of the difference of F1 scores of the pair. All learning-to-rank approaches are further described in Section 1.6.

We first discuss the dataset used for training and validating the ranking model (Section 5.4.1). Afterward, we describe the results of preliminary experiments, where we train the GPT-2-small ranking model (Section 5.4.2), and the results of final experiments where we train the GPT-2-medium ranking model (Section 5.4.3).

5.4.1 Data

A generation model is used to generate 4 candidate outputs for each of the first 30,000 utterances[3] from the training split of the ConvAI2 dataset. Out of them, the first 20,000 are used for training a ranking model (*we call them the Training-Rank dataset*) and the remaining 10,000 are used for validation (*Valid-Rank dataset*). The generation model is then used in the same way to generate candidates for the whole validation split of the ConvAI2 dataset (for clarity, in the following text we call it ConvAI2 Valid dataset). In our evaluations, we test the combination of a generation model and a ranking model, where a perfect ranking model would select the candidate with highest F1 score for each sample.

Distribution Shift

First, we use the GPT-2-medium checkpoint, which scores highest in the F1 score (discussed in Section 5.1) for the first stage, i.e., as a generator. For clarity, we use the name GPT-2-medium$_{\text{full}}$ for this model. The generation is done with diverse beam search since a perfect ranking model could reach the highest F1 scores with this decoding method (Table 5.5).

However, the GPT-2-medium$_{\text{full}}$ model is trained on the full training dataset of the ConvAI2 dataset, i.e., also on the Training-Rank and Valid-Rank datasets. This causes a shift of distribution of F1 scores between the Training-Rank and Valid-Rank datasets on one side and the ConvAI2 Valid dataset on the other side, e.g., oracle maxima and minima are about two points higher for Training-Rank and Valid-Rank datasets which were seen by the model during training (see Table 5.9).

Dataset	F1 score			
	Max	Min	NLL-Selected	Random
GPT-2-medium$_{\text{full}}$ Training-Rank	30.12%	12.68%	22.06%	20.97%
GPT-2-medium$_{\text{full}}$ Valid-Rank	30.80%	13.02%	22.66%	21.54%
GPT-2-medium$_{\text{full}}$ ConvAI2 Valid	27.94%	12.27%	20.36%	19.63%
GPT-2-medium $_{\text{-30k}}$ Training-Rank	27.93%	11.70%	20.04%	19.30%
GPT-2-medium $_{\text{-30k}}$ Valid-Rank	28.15%	11.80%	20.30%	19.55%
GPT-2-medium $_{\text{-30k}}$ ConvAI2 Valid	27.44%	11.46%	19.63%	18.95%

Table 5.9: F1 score potential for candidates generated by the GPT-2-medium model (discussed in Section 5.1) and the GPT-2-medium$_{\text{-30k}}$ model with diverse beam search with beam size 4.

[3]We choose the size of 30,000 utterances since it offers a reasonable ratio between speed of experimenting and validation performance. With our code, the speed of training is about 10 minutes per epoch.

Therefore we train another generation model, the GPT-2-medium $_{\text{-30k}}$ with the same hyper-parameters using all but the first 30,000 utterances from the training split of the dataset. Thanks to this retraining, there is no significant F1 score distribution shift between the candidates generated by GPT-2-medium $_{\text{-30k}}$ on Training-Rank, Valid-Rank and ConvAI2 Valid data; however, we see a performance drop of about 0.8 points compared to generations by the GPT-2-medium model trained on the full dataset (Table 5.9). We conduct the preliminary experiments on generations by the GPT-2-medium$_{\text{full}}$ model, and the final experiment on generations by both models.

Training Dataset for Pairwise Methods The pairwise ranking methods are trained on pairs of examples (Section 1.6.2). Burges et al. [2005] train the RankNet model only on pairs whose target scores differ more than a chosen threshold. We follow this approach and train a ranking model on all pairs of generations by the selected models whose F1 scores differ by more than 3%.[4]

5.4.2 Results of Preliminary Experiments

In this section we discuss results of preliminary ranking experiments, which are done with GPT-2-small ranking models operating on generations by the GPT-2-medium$_{\text{full}}$ model (see Section 5.4.1).

All ranking methods trained in this section improve the performance of the two-stage system. Similarly to results of Burges et al. [2005] and Burges et al. [2006], the performance of both pairwise methods exceeds the performance of the pointwise method. However, while the ranking models improve the resulting F1 score, the gap to the oracle maximum (i.e., a ranker that always selects the candidate with the highest F1 score) is still large (7 points), and the improvements compared to a single-stage generation are small.

Decoding Method	F1
Oracle	27.94%
Diverse Beam Search, beam size 4	20.36%
Beam Search, beam size 2	20.57%
Pointwise Ranker (GPT-2-small)	20.64%
Pairwise RankNet Ranker (GPT-2-small)	20.87%
Pairwise LambdaRank Ranker (GPT-2-small)	**20.94%**

Table 5.10: Comparison of different decoding methods on ConvAI2 Valid data. Oracle, and rankers all operate on candidates generated by the GPT-2-medium$_{\text{full}}$ model (see Section 5.4.1).

5.4.3 LambdaRank with GPT-2-medium

Based on the results of experiments in Section 5.4.2, we conclude that LambdaRank is the most promising method on our data. In this section, we train the

[4]This number was selected with an overfitting experiment, i.e., overfitting a dataset of only 100 examples was fastest. Removing all samples where there is no pair of generations whose F1 scores differ by more than 3% shrinks the dataset by 15%.

GPT-2-medium LambdaRank ranking model on the response candidates generated by the GPT-2-medium $_{\text{-30k}}$ model and compare the performance of combinations of three generator models (GPT-2-medium$_{\text{full}}$, GPT-2-medium $_{\text{-30k}}$, LMEDR [Chen et al., 2023]) and two ranking models (GPT-2-small LambdaRank and GPT-2-medium LambdaRank).

Method	F1
GPT-2-medium$_{\text{full}}$ (diverse beam search)	20.36%
GPT-2-medium$_{\text{full}}$ (diverse beam search oracle)	27.94%
GPT-2-medium$_{\text{full}}$ + GPT-2-small LambdaRank	20.94%
GPT-2-medium$_{\text{full}}$ + GPT-2-medium LambdaRank	21.13%
GPT-2-medium $_{\text{-30k}}$ (diverse beam search)	19.63%
GPT-2-medium $_{\text{-30k}}$ (diverse beam search oracle)	27.44%
GPT-2-medium $_{\text{-30k}}$ + GPT-2-small LambdaRank	20.18%
GPT-2-medium $_{\text{-30k}}$ + GPT-2-medium LambdaRank	20.32%
LMEDR (diverse beam search)	22.05%
LMEDR (diverse beam search oracle)	29.55%
LMEDR + GPT-2-small LambdaRank	22.08%
LMEDR + GPT-2-medium LambdaRank	22.34%

Table 5.11: The comparison of F1 scores of single-stage generators and GPT-2-small and GPT-2-medium LambdaRank ranking models on the ConvAI2 Valid dataset. LMEDR refers to the state-of-the-art model by Chen et al. [2023].

In Table 5.11 we can see that for all generation models, the GPT-2-medium LambdaRank ranking model scores about 0.2 points higher than the GPT-2-small LambdaRank ranking model.

The most interesting result is that both ranking models learn features that generalize to generations produced by the LMEDR model with BART backbone [Lewis et al., 2020] (i.e., to a generation model which was pretrained on different data than the GPT-2 family of models). Both two-stage systems consisting of LMEDR generator and LambdaRank ranker outperform the plain LMEDR generator. Moreover LMEDR generator paired with GPT-2-medium LambdaRank reaches the new state of the art on ConvAI2 dataset on the F1 metric.

5.5 Direct Preference Optimization

Direct Preference Optimization (DPO, [Rafailov et al., 2023], Section 1.5.7) is an optimization algorithm for finetuning a language model to preference data. Similarly to previous section, we conduct experiments with the baseline GPT-2-medium model (Section 5.1) and the pairwise ranking dataset introduced in Section 5.4.1, where the pairs of response candidates are generated by the aforementioned GPT-2-medium model. We use an implementation of the DPO algorithm from the Huggingface `trl` library [von Werra et al., 2020].

Training Setup We find it interesting that contrary to learning rates in the range [1e-5, 1e-4] successfully used for previous experiments, the learning rates required for DPO to not diverge (range [1e-7, 1e-6]) are two orders of magnitude lower. The model needs just a few hundred steps to reach its lowest loss and

then starts overfitting. The model which reached the highest validation F1 score trained for just 4000 steps, with an aggressive learning rate decay from 5e-7 to 1e-10. In the following text, we call this model GPT-2-medium$_{\text{DPO}}$

Results Results for DPO are given in Table 5.12. Contrary to training with language modeling loss (Section 5.1), or joint training of language modeling and next utterance classification (Section 5.3), greedy search is the best performing decoding method according to F1 score for the DPO-finetuned model and beam search deteriorates the score for all tested beam sizes $\{2, 4, 8, 16\}$.

Interestingly, it seems that DPO and LambdaRank ranking are complementary, since generating candidates with GPT-2-medium$_{\text{DPO}}$ and ranking them with GPT-2-medium LambdaRank ranking model reaches the top F1 score observed with systems composed only from our models (Table 5.12), i.e., excluding LMEDR + GPT-2-medium LambdaRank discussed in Section 5.4.

Model	Decoding Method	F1
GPT-2-medium	greedy	19.8%
	beam-2	20.57%
	diverse-beam-4	20.36%
	GPT-2-medium LambdaRank (diverse-beam-4)	21.13%
GPT-2-medium$_{\text{DPO}}$	greedy	21.15%
	beam-2	20.99%
	diverse-beam-4	20.97%
	GPT-2-medium LambdaRank (diverse-beam-4)	**21.51%**

Table 5.12: Comparison of the model finetuned with DPO algorithm to models from our previous experiments.

5.6 Prompt Engineering with GPT-3.5

According to Chatbot Arena, an online large-scale human evaluation framework Chiang et al. [2024], at the time of writing, models by OpenAI (GPT-3.5, GPT-4, GPT-4o) [Ouyang et al., 2022, OpenAI, 2023] are among the top performing open-domain chat models.[5] Therefore, we decided to use these models with a simple prompting strategy, to compare to our finetuned models from previous sections.

First, we run a preliminary experiment to test the capabilities of the OpenAI models on a few dialogues from the ConvAI2 dataset. Since responses from all the models look human-like, we choose the cheapest available option, the GPT-3.5 for our experiments (which is 10-60 times cheaper per-token in comparison to other OpenAI models).

The experiment is conducted in two stages. First, we utilize a small subset of the ConvAI2 training data (1000 randomly selected samples) as a development set for prompt engineering, with the aim to improve the outputs' F1 score. Then we apply the best-performing prompt and hyper-parameters on the validation dataset and measure the final scores.

[5] As of June 22, 2024, GPT-4o is on the first place, GPT-4 models rank between 4th and 6th place and GPT-3.5 models are between 40th and 50th place out of 108 registered models.

5.6.1 Prompt Engineering on the ConvAI2 Training Data

As a baseline, we provide only the following prompt to the model: *"You are a chatbot and you have the following traits: <numbered traits>. Continue the dialogue."* Here, <numbered traits> lists the persona sentences (see Figure 5.8). We spot that the model generates responses containing 19 words on average, which is 10 more than the average number of words in the dataset (cf. Section 2.4.2).

Therefore we added an instruction to *"follow the style of the dialogue. If long sentences are used in the dialogue continue with using long sentences, if short sentences, then continue with short."* Since the model answers are decoded using multinomial sampling (see Section 1.2.4), we experience a lot of noise. Runs 2-6 in Figure 5.7 are ran with a small variations of instructions to use shorter sentences and the score varies between 17.8% and 18.5%

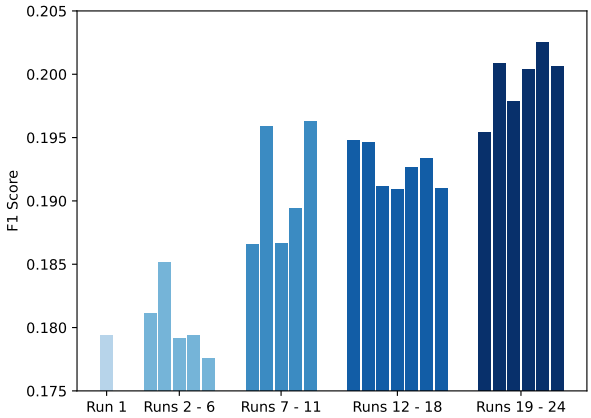


Figure 5.7: F1 scores of GPT-3.5 model on the selected development subset of ConvAI2 training data for all the prompts that we explored during the prompt engineering process.

To further improve the F1 score, we measure the precision and recall for individual words in GPT-3.5's outputs. We found few candidates with low precision (i.e., the gold answers do not contain these words, while the model's responses do) and instructed the model to *"not use words 'some', 'can', 'also', 'on', 'so' "*. This lead to an improvement of about 0.5 F1 points (18.0 → 19.1, runs 7-11). Then we examined words with low recall and instructed the model that it *"may use some of these words: 'am', 'are', 'what', 'do', 'of', 'like', 'is', 'have',"* which resulted in another small improvement in F1 score (19.1 → 19.3, runs 12-18).

To reduce the variance in model performance, we used top-p nucleus sampling (see Section 1.2.4). This not only reduced variance, but also improved the results in terms of F1 score (runs 19-24). Figure 5.8 contains the final prompt, which we use along with the top-p parameter set to 0.5.

5.6.2 Final Results

While the prompt engineering improved the F1 score performance of the GPT-3.5, the performance still lags even behind the greedy-decoding performance of the

You are a person with following traits: 1: 'you have entered into many violin competitions before and have placed in a few of them .' 2: 'you ' ve a pomeranian .' 3: 'your older sister plays clarinet .' 4: 'your mom is a music teacher at the elementary school .'. Try to follow the style of the dialogue (formal or chatty). Do not use words 'some', 'can', 'also', 'on', 'so', 'with', 'yeah', 'time'. You may use some of these words: 'am', 'are', 'what', 'do', 'of', 'like', 'is', 'have'. Use at most 14 words in your answers.

Figure 5.8: Example system prompt for sample 106528 from the training split of the ConvAI2 dataset.

baseline finetuned GPT-2-medium (Table 5.13). However, upon manual inspection, GPT-3.5's outputs are natural and consistent. We believe that this is more of an issue with the F1 metric or potentially the reference responses themselves. This further motivates us to perform human evaluation in Section 5.7.

Model	F1
GPT-3.5 - baseline prompt	18.78%
GPT-3.5 - prompt engineered	19.62%
GPT-2-medium - baseline, greedy	19.8%
GPT-2-medium$_{\text{DPO}}$ + GPT-2-medium LambdaRank	21.51%
LMEDR + GPT-2-medium LambdaRank	22.34%

Table 5.13: Comparison of our best performing models to GPT-3.5 in terms of F1 score.

5.7 Human Evaluation

As the F1 results do not conform to our manual checks for the results (and automatic metrics are known to correlate weakly with human judgements, see Section 3.1.5), we conduct a small-scale human evaluation experiment. Following the setup of ChatBot Arena [Chiang et al., 2024] (discussed in Section 3.2), we opt for single-turn ranking human evaluation, where annotators assess the overall quality of provided responses.

We conduct two experiments. First, when manually examining generations of GPT-3.5, we found that the responses generated by GPT-3.5 are more grammatical than gold responses from the dataset and do not contain other data quality issues discussed in Section 2.5. We conduct a round of human evaluation to verify whether generations by GPT-3.5 are preferable to the gold responses from the dataset (Section 5.7.1). In the second round of human evaluation, we compare the performance of three selected models (Section 5.7.2).

The human evaluation is conducted in-house through the VisuaLLM web framework [Trebuňa and Dusek, 2023]. Three annotators participate in each round. First, the evaluation task is explained to the evaluators (Figure A.1 in the appendix). During the evaluation, the evaluator is presented with a dialogue, a persona of the bot and two candidate responses generated by anonymous models (Figure A.2 in the appendix). The evaluator may either select one of the

candidate responses as preferred, or "Tie" (both responses are equally good), or "Both are bad" (neither response is adequate).

5.7.1 GPT-3.5 is Preferred to Gold Responses

To compare GPT-3.5 to human references in ConvAI2 data, we collect 155 pairwise comparisons by three annotators.[6] Pairs to be compared are randomly drawn with replacement from the validation dataset. The results of this evaluation confirm our intuition. Counting ties as half point and discarding bad ties, the generations of the GPT-3.5 model are preferred in 63.3% of pairs (see details in Figure 5.9).

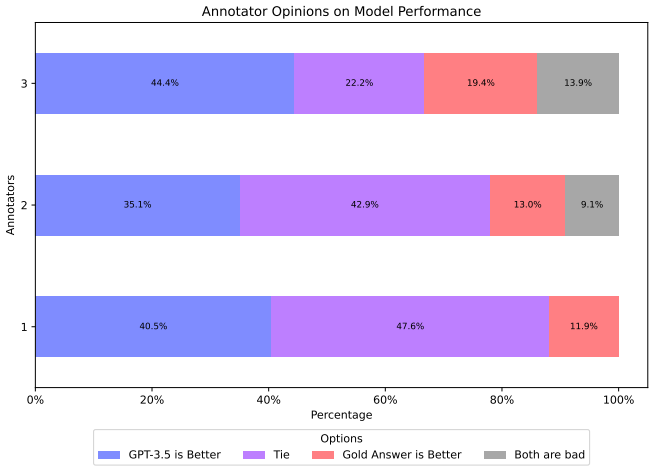


Figure 5.9: Annotator preference between human references and GPT-3.5-generated answers on the validation split of the ConvAI2 dataset.

5.7.2 Human Evaluation Results

We use the same setting as in previous section for comparing the quality of our model outputs. We collect 352 pairwise comparisons by three annotators.[7]

Generations of three models are compared, namely the prompt-engineered GPT-3.5 (see Section 5.6), our best model according to the F1 score (see Section 5.5) and the state-of-the-art LMEDR + GPT-2-medium LambdaRank (Sections 4.4, 5.4). Again, the pairs to be compared are randomly drawn with replacement from the validation dataset. We present the results as a heat-map of win rates of these models (Figure 5.10).

We compare the ordering induced by validation F1 scores and the ordering induced by win rates according to human evaluation. We see that the GPT-3.5 model that scores the lowest on the F1 score is actually the most preferred by humans. The ordering of both models finetuned on the ConvAI2 dataset according to human evaluation stays the same as the ordering according to the F1 score (Table 5.14).

[6] 155 pairwise comparisons by three annotators in total. The annotators provided respectively 42, 77 and 36 comparisons.

[7] 352 pairwise comparisons by three annotators in total. The annotators provided respectively 139, 105 and 108 comparisons.

Model	Win Rate	F1 Score
GPT-2-medium$_{\text{DPO}}$ + GPT-2-medium LambdaRank	46%	21.51%
LMEDR + GPT-2-medium LambdaRank	58%	**22.34%**
GPT-3.5	**75%**	19.62%

Table 5.14: Win rates and F1 scores of evaluated models.

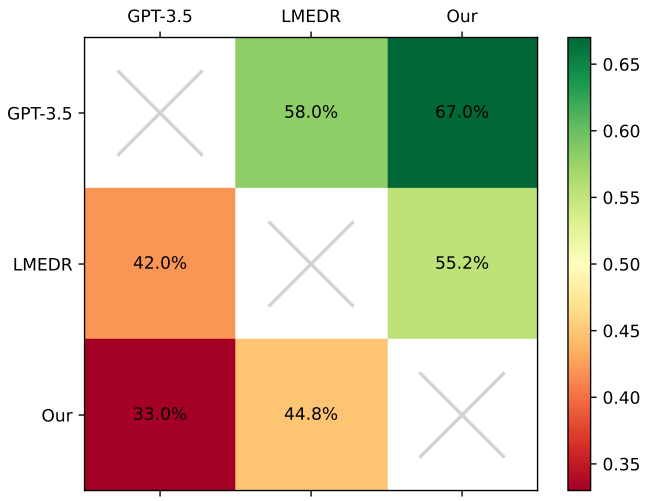


Figure 5.10: Win rates of selected models against each other.

5.8 Discussion

In this chapter we conducted several experiments on the ConvAI2 dataset [Dinan et al., 2020]. Firstly, in Sections 5.1, 5.2 we finetuned models from the GPT-2 family [Radford et al., 2019] on the ConvAI2 dataset [Dinan et al., 2020] and explored their decoding performance. Later, in Section 5.3, we explored the multi-task finetuning setup proposed by TransferTransfo [Wolf et al., 2019] and stemming from the conclusions of this section we followed the idea of a two-stage generate-and-rank approach in Section 5.4. We finetuned the GPT-2-medium model with the direct preference optimization algorithm [Rafailov et al., 2023] which showed that DPO and two-stage setup are complementary and bot lead to F1 score improvements (Section 5.5). At the end, we engineered a prompt that improves the F1 score of generations by the GPT-3.5 model, but still lags behind the previously produced finetuned models, despite the outputs appearing better quality (Section 5.6). We then conducted a small scale human evaluation to counterbalance the F1 metric (Section 5.7).

With our extensions, we were able to improve the performance of the GPT-2 model in terms of F1 score (Table 5.15) and hits@1 (Table 5.6). Our two-stage approach combined with the previous state-of-the-art LMEDR model [Chen et al., 2023] even reached the new state-of-the-art in F1 score (Table 5.11). Ultimately, despite the ranking of models based on the F1 score, the human evaluation indicated that the GPT-3.5 model outperformed the others (Figure 5.10). According to another round of human evaluation the generations by the model are even preferred to the gold responses in the dataset (Section 5.7.1), i.e., the GPT-3.5 model solves the task better than paid crowd workers.

Model	F1 Score
GPT-2-medium (greedy)	19.80%
GPT-2-medium (beam search, beam size = 2)	20.57%
GPT-2-medium (GPT-2-medium LambdaRank, diverse beam search)	21.13%
$\text{GPT-2-medium}_{\text{DPO}}$ (GPT-2-medium, diverse beam search)	21.51%

Table 5.15: F1 scores of different versions of finetuned GPT-2-medium model.

Conclusion

In this thesis, we explored end-to-end open-domain dialogue modeling on the ConvAI2 dataset [Dinan et al., 2020].

First, we established a baseline performance by finetuning models from the GPT-2 family [Radford et al., 2019] on the ConvAI2 dataset. We showed that evaluating model perplexity is non-trivial and even figures reported in peer-reviewed works may be incomparable to each other. In the subsequent experiments we focused on two metrics, the F1 score, and the hits@1 on next utterance classification. We improved the measured F1 score by exploring variants of beam search. We then further extend the setup following our research questions:

Can we improve generation using a two-stage setup with ranking? We showed that the default ranking of candidates based on length-normalized negative log likelihood is not optimal, and we explored a two-stage approach where a ranking model is used to rank multiple generated-candidate responses. First, following Wolf et al. [2019], we added a ranking head (i.e., a linear layer with sigmoid activation) to the GPT-2-medium model and trained it for the next utterance classification objective. While this lead to an improvement in the performance on hits@1 on next utterance classification, it did not improve the generation performance as measured by the F1 score. We then trained separate ranking models by following learning-to-rank approaches [Burges et al., 2006]. We showed that a two-stage system where the GPT-2-medium generates several candidate responses and a separately trained GPT-2-medium LambdaRank ranking model ranks these responses, improves the performance as measured by the F1 score. When using the GPT-2-medium LambdaRank ranking model to rank candidate responses generated by the state-of-the-art LMEDR model [Chen et al., 2023], we even reached a new state-of-the-art F1 score on this task.

Can we improve generation using direct preference optimization? We then used direct preference optimization (DPO, [Rafailov et al., 2023]) to further finetune the GPT-2-medium generation model. We showed that the DPO and LambdaRank ranking are complementary since a two-stage system where GPT-2-medium finetuned with DPO generates candidate responses which are ranked by GPT-2-medium trained with LambdaRank reaches the top F1 score observed with systems composed only from our own models.

How does finetuning compare to prompting LLMs? While finetuning language models is a common method of improving performance of a pretrained model on a downstream task, prompting approaches [Liu et al., 2023] seem to be a viable alternative. We engineer a prompt that improves the F1 score of the GPT-3.5 large language model [Ouyang et al., 2022, OpenAI, 2023] on the ConvAI2 dataset. While GPT-3.5 under-performs even our simplest baseline in terms of F1 score, upon manual checking of the generated responses, we found out that we actually prefer the responses generated by GPT-3.5 to the responses generated by other models, and also to the gold responses from the dataset. We

conducted a small-scale human evaluation, which confirmed that the GPT-3.5-generated responses are preferred by the annotators compared to our models, but also compared to the gold responses from the dataset.

Future Work There are several interesting extensions to our work, which we did not follow due to shortage of time. Due to memory constraints of available GPUs, the GPT-2-large model was the largest model that we trained. However, with LoRA [Hu et al., 2022], and QLoRA [Dettmers et al., 2023] approaches (or even with prompt tuning approaches described in Section 1.5.4) we could be able to train larger and hence potentially better-performing models, such as LLama-2 [Touvron et al., 2023], Qwen [Bai et al., 2023], or other open-source large language models.

We only used the simplest learning-to-rank methods to train ranking models. The list-wise methods mentioned in Section 1.6 are a natural extension to our experiments. Experiments with a larger number of generated candidates (i.e., generating more than four candidates), and candidates generated by other decoding methods are another natural extension to our work.

Lastly, we only used a single iteration of DPO finetuning. An approach where a model finetuned by DPO would generate more preference data, which would in turn be used for another round of DPO finetuning, is an experiment that we would have definitely tried if not for the shortage of time.

Bibliography

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. Towards a human-like open-domain chatbot, 2020. URL https://arxiv.org/abs/2001.09977.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Jade Goldstein, Alon Lavie, Chin-Yew Lin, and Clare Voss, editors, *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://aclanthology.org/W05-0909.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL https://api.semanticscholar.org/CorpusID:11212020.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023. URL https://arxiv.org/abs/2309.16609.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, mar 2003. ISSN 1532-4435.

John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In Annie Zaenen and Antal van den Bosch, editors, *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/P07-1056.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051. URL https://aclanthology.org/Q17-1010.

Léon Bottou. *On-line learning and stochastic approximations*, page 9–42. Cambridge University Press, USA, 1999. ISBN 0521652634.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 89–96, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102363. URL https://doi.org/10.1145/1102351.1102363.

Christopher Burges, Robert Ragno, and Quoc Le. Learning to rank with nonsmooth cost functions. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006. URL https://proceedings.neurips.cc/paper_files/paper/2006/file/af44c4c56f385c43f2529f9b1b018f6a-Paper.pdf.

Christopher J. C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical report, Microsoft Research, 2010. URL http://research.microsoft.com/en-us/um/people/cburges/tech_reports/MSR-TR-2010-82.pdf.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 129–136, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273513. URL https://doi.org/10.1145/1273496.1273513.

Rich Caruana, Shumeet Baluja, and Tom Mitchell. Using the future to "sort out" the present: Rankprop and multitask learning for medical risk evaluation. In D. Touretzky, M.C. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1995. URL https://proceedings.neurips.cc/paper_files/paper/1995/file/36a16a2505369e0c922b6ea7a23a56d2-Paper.pdf.

Ashwini Challa, Kartikeya Upasani, Anusha Balakrishnan, and Rajen Subba. Generate, filter, and rank: Grammaticality classification for production-ready NLG systems. In Anastassia Loukina, Michelle Morales, and Rohit Kumar, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,*

Volume 2 (Industry Papers), pages 214–225, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-2027. URL https://aclanthology.org/N19-2027.

Ruijun Chen, Jin Wang, Liang-Chih Yu, and Xuejie Zhang. Learning to memorize entailment and discourse relations for persona-consistent dialogues. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. ISBN 978-1-57735-880-0. doi: 10.1609/aaai.v37i11.26489. URL https://doi.org/10.1609/aaai.v37i11.26489.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost, 2016. URL https://arxiv.org/abs/1604.06174.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference, 2024.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In Dekai Wu, Marine Carpuat, Xavier Carreras, and Eva Maria Vecchi, editors, *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012. URL https://aclanthology.org/W14-4012.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024. URL http://jmlr.org/papers/v25/23-0870.html.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL https://aclanthology.org/2020.acl-main.747.

Kevin Crowston. Amazon mechanical turk: A research tool for organizations and information systems scholars. In Anol Bhattacherjee and Brian Fitzgerald,

editors, *Shaping the Future of ICT Research. Methods and Approaches*, pages 210–221, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-35142-6.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/7137debd45ae4d0ab9aa953017286b20-Paper.pdf.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022a. URL https://openreview.net/forum?id=dXiGWqBoxaD.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022b. URL https://openreview.net/forum?id=shpkpVXzo3h.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. URL https://arxiv.org/abs/2305.14314.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=r1l73iRqKm.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, Shrimai Prabhumoye, Alan W. Black, Alexander Rudnicky, Jason Williams, Joelle Pineau, Mikhail Burtsev, and Jason Weston. The second conversational intelligence challenge (convai2). In Sergio Escalera and Ralf Herbrich, editors, *The NeurIPS '18 Competition*, pages 187–208, Cham, 2020. Springer International Publishing. ISBN 978-3-030-29135-8.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. URL http://jmlr.org/papers/v12/duchi11a.html.

Ondřej Dušek and Zdeněk Kasner. Evaluating semantic accuracy of data-to-text generation with natural language inference. In Brian Davis, Yvette Graham, John Kelleher, and Yaji Sripada, editors, *Proceedings of the 13th International Conference on Natural Language Generation*, pages 131–137, Dublin, Ireland, December 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.inlg-1.19. URL https://aclanthology.org/2020.inlg-1.19.

Nouha Dziri, Ehsan Kamalloo, Kory Mathewson, and Osmar Zaiane. Evaluating coherence in dialogue systems using entailment. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3806–3812, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1381. URL https://aclanthology.org/N19-1381.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1045. URL https://aclanthology.org/D18-1045.

Johannes Fürnkranz and Eyke Hüllermeier. *Preference Learning*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010. ISBN 3642141242.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1243–1252. JMLR.org, 2017.

Ross Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. doi: 10.1109/ICCV.2015.169.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Alex Graves. Sequence transduction with recurrent neural networks, 2012. URL https://arxiv.org/abs/1211.3711.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE, June 2016. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL https://aclanthology.org/P18-1031.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In Brian Davis, Yvette Graham, John Kelleher, and Yaji Sripada, editors, *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182, Dublin, Ireland, December 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.inlg-1.23. URL https://aclanthology.org/2020.inlg-1.23.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Fred Jelinek, Robert L Mercer, Lalit R Bahl, and James K Baker. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63, 1977.

Dan Jurafsky and James H. Martin. Speech and Language Processing. https://web.stanford.edu/~jurafsky/slp3/, 2024. [Accessed 24-06-2024].

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#KingmaB14.

John F. Kolen and Stefan C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*, pages 237–243. Wiley-IEEE Press, 2001. doi: 10.1109/9780470544037.ch14.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL https://aclanthology.org/D18-2012.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243.

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

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California, June 2016a. Association for Computational Linguistics. doi: 10.18653/v1/N16-1082. URL https://aclanthology.org/N16-1082.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California, June 2016b. Association for Computational Linguistics. doi: 10.18653/v1/N16-1014. URL https://aclanthology.org/N16-1014.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 994–1003, Berlin, Germany, August 2016c. Association for Computational Linguistics. doi: 10.18653/v1/P16-1094. URL https://aclanthology.org/P16-1094.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9), jan 2023. ISSN 0360-0300. doi: 10.1145/3560815. URL https://doi.org/10.1145/3560815.

Qian Liu, Yihong Chen, Bei Chen, Jian-Guang Lou, Zixuan Chen, Bin Zhou, and Dongmei Zhang. You impress me: Dialogue generation via mutual persona perception. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1417–1427, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.131. URL https://aclanthology.org/2020.acl-main.131.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019. URL https://api.semanticscholar.org/CorpusID:198953378.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. The flan collection: designing data and methods for effective instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In Alexander Koller, Gabriel Skantze, Filip Jurcicek, Masahiro Araki, and Carolyn Penstein Rose, editors, *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, Prague, Czech Republic, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-4640. URL https://aclanthology.org/W15-4640.

Ryan Lowe, Iulian Vlad Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. On the evaluation of dialogue systems with next utterance classification. In Raquel Fernandez, Wolfgang Minker, Giuseppe Carenini, Ryuichiro Higashinaka, Ron Artstein, and Alesia Gainer, editors, *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 264–269, Los Angeles, September 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-3634. URL https://aclanthology.org/W16-3634.

Li Lucy and David Bamman. Gender and representation bias in GPT-3 generated stories. In Nader Akoury, Faeze Brahman, Snigdha Chaturvedi, Elizabeth Clark, Mohit Iyyer, and Lara J. Martin, editors, *Proceedings of the Third Workshop on Narrative Understanding*, pages 48–55, Virtual, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.nuse-1.5. URL https://aclanthology.org/2021.nuse-1.5.

Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. URL https://aclanthology.org/D15-1166.

Andrea Madotto, Zhaojiang Lin, Chien-Sheng Wu, and Pascale Fung. Personalizing dialogue agents via meta-learning. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5459, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1542. URL https://aclanthology.org/P19-1542.

Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, page 165–172, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450324090. doi: 10.1145/2507157.2507163. URL https://doi.org/10.1145/2507157.2507163.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering, 2018. URL https://arxiv.org/abs/1806.08730.

Shikib Mehri and Maxine Eskenazi. USR: An unsupervised and reference free evaluation metric for dialog generation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 681–707, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.64. URL https://aclanthology.org/2020.acl-main.64.

John Mendonça, Patrícia Pereira, Helena Moniz, Joao Paulo Carvalho, Alon Lavie, and Isabel Trancoso. Simple LLM prompting is state-of-the-art for robust and multilingual dialogue evaluation. In Yun-Nung Chen, Paul Crook,

Michel Galley, Sarik Ghazarian, Chulaka Gunasekara, Raghav Gupta, Behnam Hedayatnia, Satwik Kottur, Seungwhan Moon, and Chen Zhang, editors, *Proceedings of The Eleventh Dialog System Technology Challenge*, pages 133–143, Prague, Czech Republic, September 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.dstc-1.16.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

T. Mikolov, Martin Karafiát, Lukas Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. *Proceedings of Interspeech*, 2, 01 2010.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. Why we need new evaluation metrics for NLG. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1238. URL https://aclanthology.org/D17-1238.

OpenAI. Gpt-4 technical report, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.

Mark Palatucci, Dean Pomerleau, Geoffrey E Hinton, and Tom M Mitchell. Zero-shot learning with semantic output codes. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper_files/paper/2009/file/1543843a4723ed2ab08e18053ae6dc5b-Paper.pdf.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In Kevin Knight, Hwee Tou Ng, and Kemal Oflazer, editors, *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219855. URL https://aclanthology.org/P05-1015.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5): 1–17, 1964. ISSN 0041-5553. doi: https://doi.org/10.1016/0041-5553(64)90137-5. URL https://www.sciencedirect.com/science/article/pii/0041555364901375.

Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina, editors, *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3049. URL https://aclanthology.org/W15-3049.

Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10208–10219, 2021. doi: 10.1109/CVPR46437.2021.01008.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training, 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=HPuSIXJaa9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the lim-

its of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), jan 2020. ISSN 1532-4435.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. Towards empathetic open-domain conversation models: A new benchmark and dataset. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1534. URL https://aclanthology.org/P19-1534.

Mario Rodríguez-Cantelar, Chen Zhang, Chengguang Tang, Ke Shi, Sarik Ghazarian, João Sedoc, Luis Fernando D'Haro, and Alexander I. Rudnicky. Overview of robust and multilingual automatic evaluation metricsfor open-domain dialogue systems at DSTC 11 track 4. In Yun-Nung Chen, Paul Crook, Michel Galley, Sarik Ghazarian, Chulaka Gunasekara, Raghav Gupta, Behnam Hedayatnia, Satwik Kottur, Seungwhan Moon, and Chen Zhang, editors, *Proceedings of The Eleventh Dialog System Technology Challenge*, pages 260–273, Prague, Czech Republic, September 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.dstc-1.28.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

Ananya B. Sai, Mithun Das Gupta, Mitesh M. Khapra, and Mukundhan Srinivasan. Re-evaluating adem: a deeper look at scoring dialogue responses. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33016220. URL https://doi.org/10.1609/aaai.v33i01.33016220.

Sashank Santhanam and Samira Shaikh. Towards best experiment design for evaluating dialogue system output. In Kees van Deemter, Chenghua Lin, and Hiroya Takamura, editors, *Proceedings of the 12th International Conference on Natural Language Generation*, pages 88–94, Tokyo, Japan, October–November 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-8610. URL https://aclanthology.org/W19-8610.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2019.

Kurt Shuster, Da Ju, Stephen Roller, Emily Dinan, Y-Lan Boureau, and Jason Weston. The dialogue dodecathlon: Open-domain knowledge and image grounded conversational agents. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2453–2470, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.222. URL https://aclanthology.org/2020.acl-main.222.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.1556.

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

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/sutskever13.html.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. RUBER: an unsupervised method for automatic evaluation of open-domain dialog systems. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th*

innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 722–729. AAAI Press, 2018. doi: 10.1609/AAAI.V32I1.11321. URL https://doi.org/10.1609/aaai.v32i1.11321.

Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, page 77–86, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781595939272. doi: 10.1145/1341531.1341544. URL https://doi.org/10.1145/1341531.1341544.

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, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen 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 Chi, and Quoc Le. Lamda: Language models for dialog applications, 2022. URL https://arxiv.org/abs/2201.08239.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

František Trebuňa and Ondrej Dusek. VisuaLLM: Easy web-based visualization for neural language generation. In C. Maria Keet, Hung-Yi Lee, and Sina Zarrieß, editors, *Proceedings of the 16th International Natural Language Generation Conference: System Demonstrations*, pages 6–8, Prague,

Czechia, September 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.inlg-demos.3.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R. Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models, 2018. URL https://arxiv.org/abs/1610.02424.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl, 2020.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=gEZrGCozdqR.

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 Isaac, Julia Haas, Sean Legassick, Geoffrey Irving, and Iason Gabriel. Taxonomy of risks posed by language models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 214–229, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450393522. doi: 10.1145/3531146.3533088. URL https://doi.org/10.1145/3531146.3533088.

Karl Weiss, Taghi Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3, 05 2016. doi: 10.1186/s40537-016-0043-6.

Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. Dialogue natural language inference. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3731–3741, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1363. URL https://aclanthology.org/P19-1363.

Jeremy West, Dan Ventura, and Sean Warnick. Spring research presentation: A theoretical foundation for inductive transfer. *Brigham Young University, College of Physical and Mathematical Sciences*, 1(08), 2007.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, may 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. Transfertransfo: A transfer learning approach for neural network based conversational agents, 2019. URL https://arxiv.org/abs/1901.08149.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Xiaoxia Wu, Cheng Li, Reza Yazdani Aminabadi, Zhewei Yao, and Yuxiong He. Understanding int4 quantization for language models: latency speedup, composability, and failure cases. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Jing Xu, Arthur Szlam, and Jason Weston. Beyond goldfish memory: Long-term open-domain conversation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5180–5197, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.356. URL https://aclanthology.org/2022.acl-long.356.

Yilin Yang, Liang Huang, and Mingbo Ma. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3054–3059, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1342. URL https://aclanthology.org/D18-1342.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: generalized autoregressive pretraining for language

understanding. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1205. URL `https://aclanthology.org/P18-1205`.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. DIALOGPT : Large-scale generative pre-training for conversational response generation. In Asli Celikyilmaz and Tsung-Hsien Wen, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.30. URL `https://aclanthology.org/2020.acl-demos.30`.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1061. URL `https://aclanthology.org/P17-1061`.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, 2015. doi: 10.1109/ICCV.2015.11.

List of Figures

List of Tables

A. Attachments

A.1 Human Evaluation

In this section we show the instructions presented to each annotator at the beginning of human evaluation (Figure A.1), and the screenshot of the evaluators screen (Figure A.2).

Welcome to the Human Evaluation of Generations on the ConvAI2 dataset.

ConvAI2 dataset contains samples, each consisting of a dialogue between two persons who meet each other for the first time.

Each sample from the dataset consists of a dialogue and a description of the persona the bot represents. For example, the bot may represent a persona who identifies with the following trait: "I love cooking"

The bot solves the task of so-called *next utterance prediction*. In this task, the bot has access to the dialogue history (what was said by each interlocutor) and the description of the bot's personality. The bot then generates an answer to the last utterance produced by its dialogue partner.

You will help us evaluate the individual bots. You will be presented with two tables; in one table, there will be the description of the bot's personality, and in the other table, there will be the dialogue history.

You will see two generations by two different bots at the bottom of the page, and you will have 4 options. You can either select the generation of the first model, of the second model, or that both generations are equally good (this option is named "`Tie`"), or that both generations are similarly bad (this option is named "`Both are bad`").

Figure A.1: Instructions presented to the evaluators.

Figure A.2: Screenshot of the evaluators screen.

A.2 User Documentation for Attached Code

In the attachment to this thesis, we include the code that was used for our experimentation as well as `csv` files with generations of selected models. The code is split into parts which correspond to individual experiments. In the following sections we show the instructions to run the dataset browser (Section A.2.1), to conduct finetuning experiments (Section A.2.2).

A.2.1 Exploration

The user can use the command in Figure A.3 to browse through the ConvAI2 dataset [Dinan et al., 2020]. The browser is based on the VisuaLLM library [Trebuňa and Dusek, 2023] and a screenshot of the browser is in Figure A.4.

```
flask --app src.experiments.explore.app run
```

Figure A.3: Command that starts the backend server with a dataset browser.

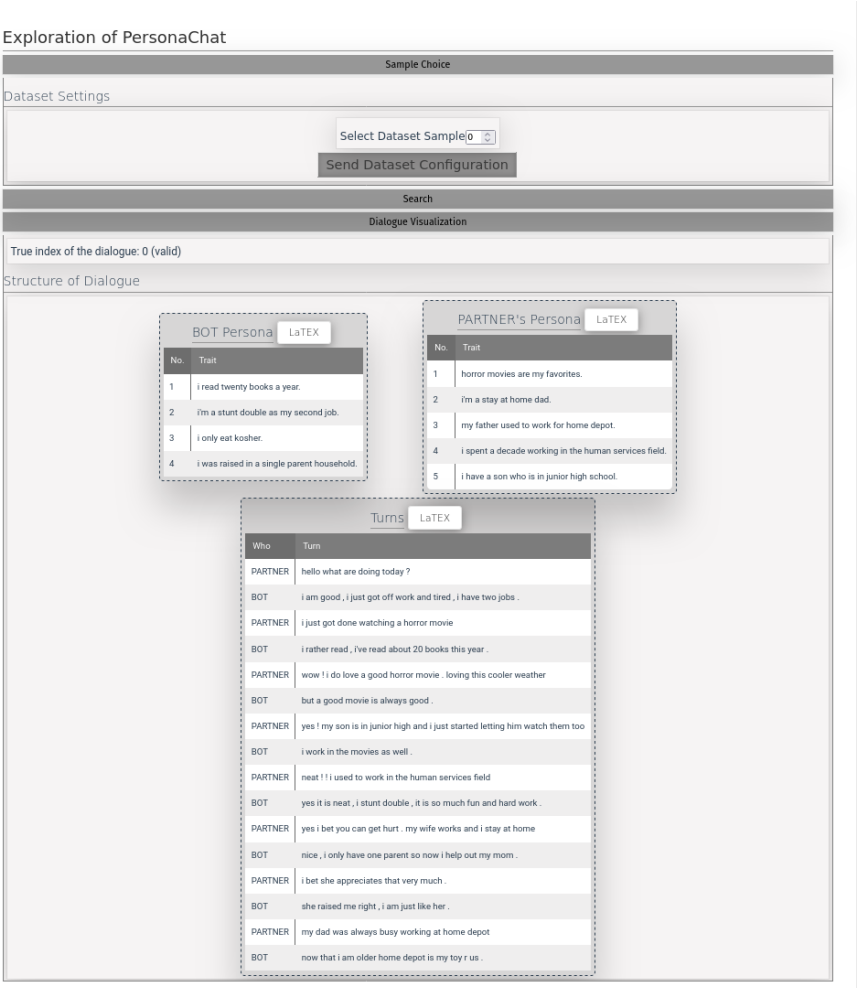


Figure A.4: Screenshot of the dataset browser screen

A.2.2 Experiments

Code for all the experiments discussed in Chapter 5 is written ad-hoc., i.e., the hyper-parameters are hard-coded and to use the code a thorough inspection is needed. In the following files, the user may find commands and documentation for the conducted experiments:

- Finetuning experiments: `src/experiments/experiment_training/README.md`
- Prompting experiments: `src/experiments/chat_gpt_prompting/README.md`

Human Evaluation

In order to run human evaluation in the setup described in Section 5.7, the user can run the command in Figure A.5. In the attachment, we also include the `csv` tables with collected human annotations. These can be found in:

- `src/experiments/human_evaluation_comparing/data_chat_gpt_better_than_humans`

  This directory contains generations of GPT-3.5 as well as collected pairwise comparisons for the experiment described in Section 5.7.1.

- `src/experiments/human_evaluation_comparing/data_model_comparison`

  This directory contains generations of three models selected for human evaluation as well as collected pairwise comparisons for the experiment described in Section 5.7.2.

```
flask --app src.experiments.human_evaluation_comparing.app run
```

Figure A.5: Command that starts the backend server with an app in which annotators evaluate pairs of anonymous generations.