Towards a Persona Aware Conversational Agent

Gonçalo Nuno Ventura de Melo

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisors: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Prof. André Filipe Torres Martins

Examination Committee

Chairperson: Prof. Paolo Romano
Supervisor: Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Member of the Committee: Dr. José David Lopes

November 2021

Acknowledgments

I would like to thank my parents, Fátima Ventura and José Paulo Melo for their friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible. I would also like to thank my grandparents, aunts, uncles and cousins for their understanding and support throughout all these years.

I would also like to acknowledge my dissertation supervisors Prof. Luísa Coheur and Prof. André Martins for their insight, support and sharing of knowledge that has made this thesis possible.

Thanks to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. A special shoutout to Maria Pereira, Rui Nóbrega and Carlos Sequeira, you were there for me since the first year, and I owe you more than I can ever repay (but I will try); to Miguel Ramos for the his friendship and to the rest of the ”Manucos”, João Clemente, Miguel Diniz, João Diniz, Guilherme Diniz and Zé for all the game nights, those were some of the most fun I had and I'm looking forward for all the ones to come; To João Lopes, for all the cigarette pauses and long phone conversations; and to Manuel Ramos, my longest standing friendship, who has been always with me through this whole journey.

Last but not least, I would like to thank my girlfriend, Mariana Saraiva, for the love, patiente, companionship, support and motivating me to work on this project during the times I did not feel like it.

To each and every one of you – Thank you.

Abstract

Dialog systems have been at the center of Natural Language Processing (NLP) since its inception. With a wide range of applications, this type of system is particularly interesting as a user interface, creating the possibility of a more natural and convenient user experience. In the context of Customer Support, goal-oriented dialog systems are now widely used, helping users carry out specific tasks. Traditionally, these systems were created by employing knowledge-based architectures. However, the growth of Deep Learning and the increase in data availability facilitated the development of neural dialog systems, which can be trained end-to-end. A well-known example of such a system is the “Transformer”, a self-attentional model that has achieved state-of-the-art results in multiple NLP tasks. Notwithstanding, these systems still present some shortcomings, particularly in terms of scalability. The need for large amounts of data and considerable computing power can be an impediment, especially in situations where multiple entities must be represented. In Goal-Oriented Dialog Systems, this becomes evident when considering multi-brand Customer Support, since each brand must communicate differently with its users, meaning one model must be developed and maintained for each brand. In Open-Domain System, an analogous problem arises when considering settings where multiple characters must be impersonated.

In this work, we explore how we can create conversational agents that tackle this issue, in both settings. To this end, we adapt and experiment with multiple state-of-the-art architectures together with recent datasets.

Keywords

Adapter; Artificial Intelligence; Customer Support; Deep Learning; Dialog system; Natural Language Processing; Transformer

Resumo

Os Sistemas de Diálogo estão no centro do Natural Language Processing (NLP) desde a incepção. Com enorme variedade de aplicações, este tipo de sistemas é particularmente interessante como interface, dando a hipótese de uma experiência mais natural e conveniente ao utilizador. No contexto de Apoio ao Cliente, sistemas de diálogo *goal-oriented* são amplamente usados, ajudando os utilizadores a completar tarefas específicas. Tradicionalmente, estes sistemas eram criados a partir de de arquiteturas *knowledge-based*. No entanto, o crescimento do *Deep Learning* e a disponibilidade de dados facilitou o desenvolvimento de sistemas de diálogo neuronais, que podem ser treinados *end-to-end*. Um exemplo deste tipo de sistemas é o *Transformer*, um modelo baseado em auto-atenção que atingiu resultados estado da arte em múltiplas tarefas de Processamento de Língua Natural. Apesar de tudo, estes sistemas ainda apresentam algumas lacunas, particularmente em termos de escalabilidade. A necessidade de grandes quantidades de dados e poder computacional considerável pode ser um impedimento, especialmente em situações onde múltiplas entidades devem ser representadas. Em Sistemas *Goal-Oriented*, esta limitação torna-se mais evidente, considerado Apoio ao Cliente multi-marca, visto que cada marca deve comunicar de forma diferente com os seus utilizadores, o que significa que um modelo deve ser desenvolvido e mantido para cada marca. Em sistemas *Open-Domain*, um problema análogo surge quando são consideradas situações em que múltiplas personagens devem ser representadas.

Neste trabalho, exploramos de que forma podemos criar agentes conversacionais que enfrentam este problema, em ambos os tipos de situações. Com este objetivo, experimentamos e adaptamos várias arquiteturas estado da arte, juntamente com os *datasets* mais recentes.

Palavras Chave

Adapter; Apoio ao Cliente, Aprendizagem Profunda; Inteligência Artificial; Processamento de Linguagem

Natural; Sistema de Diálogo; Transformer

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
BPE	Byte Pair Encoding
CNN	Convolutional Neural Network
FFNN	Feed-Forward Neural Network
LM	Language Model
LSTM	Long Short-Term Memory
NLL	Negative Log Likelihood
NLP	Natural Language Processing
RNN	Recurrent Neural Network
Seq2Seq	Sequence to Sequence
ReLU	Rectified Linear Unit

1 Introduction

Contents

The modeling of dialog systems is a fundamental problem in Natural Language Processing (NLP). In essence, Dialog Systems reflect our ability to artificially generate human-level conversations, bringing us closer to achieve one of the most ambitious long-term goals of AI since its advent, the Turing Test [Turing, 1950].

This type of system can have a wide range of applications, particularly as human-machine interfaces [Lowe et al., 2015] [Serban et al., 2015]. One of the most prominent areas where this can be leveraged is Customer Service Automation through the development of task-oriented dialog generation systems. Task-oriented dialog systems or task-oriented chatbots are devised to help users carry out specific tasks, such as booking plane tickets or making a reservation. This type of chatbot is typically restricted in the range of tasks it can help the user with and is optimized to make the execution of the task as fast and seamless as possible. This helps to create a more natural and convenient user experience.

Traditionally, the creation of such systems resorted to knowledge-based architectures [Weizenbaum, 1966] [Wallace, 2009]. Then, the advent of Deep Learning, coupled with the growing data abundance, allowed the development of neural dialog systems. Contrarily to the traditional models, the latter can be trained end-to-end [Chen et al., 2017]. Furthermore, sequence modeling methods can also be used to train task-oriented dialog systems. These methods receive a sequence as an input and predict a sequence as output. In Customer Support, for example, the input could be a query from a user, and the predicted sequence could be the answer to that query. More recently, new architectures based solely on attention mechanisms [Vaswani et al., 2017], neglecting recurrence and convolution, emerged as a new paradigm for sequence modeling tasks. These models have achieved state-of-the-art performance in multiple NLP tasks, such as Machine Translation [Vaswani et al., 2017]. Despite the encouraging results, they pose some problems. Namely, they depend upon large amounts of data, have high model complexity, and require considerable computing power [Kaplan et al., 2020]. These key factors render their scaling burdensome.

This is a problem in both Open and Closed Domain settings. However, for Goal-Oriented systems (Closed Domain setting), the described problems can be aggravated, considering multi-brand companies. Usually, the different brands employ a different Brand Communication Style and answer differently to a given query. This means that, in order to leverage dialog systems, such companies need to develop one model for each brand, significantly increasing the amount of data, storage space and computational power required. In Open-Domain, the same is true for settings where multiple characters must be impersonated by the system. Once again, to leverage dialog systems, one would need to develop one model for each character, raising the same issues.

The problem however is not limited to the scale. How to ensure that a neural conversational agent maintains a consistent and coherent communication style or represents truthfully a given character is still an open problem and solving it will allow us to create more engaging conversational agents.

1.1 Objective

The objective of this work is to **build a persona-aware model** that can represent multiple personas, when prompted to. As defined in [Li et al., 2016], a persona can be defined as “the character that an artificial agent, as actor, plays or performs during conversational interactions.” This means a persona can include different elements of identity such as background facts, language behaviour and even interaction style. This persona can be a brand (for Customer Support) or a character (for chit-chat). The objective of the agent is: take an input that includes the query and persona metadata, and responds as the persona is expected to. By doing this, the costs associated with employing chatbots are drastically reduced in these contexts, when employing the developed models.

1.2 Contributions

To achieve this, we make use of modern NLP methods and techniques that have showed encouraging results in Dialog Generation and adapt them to our specific task, which, to the best of our knowledge, has not been properly explored. We started by collecting/adapting two different datasets, one for Open-Domain: the Friends Corpus [Zhou and Choi, 2018]; and one for Customer Support: Twitter Customer Support Dataset [1]. To facilitate the pre-processing of these datasets we also developed a tool called “Piertotum”. This tool automates the setup of a preprocessing pipeline, speeding up the process of adapting the same datasets to different models and of experimenting with different preprocessing techniques. Then, we develop, train and test two different models: One based on the TransferTransfo model [Wolf et al., 2019], and a second one similar to it but with a key architectural difference: the use of Adapters [Houlsby et al., 2019]. Finally, to evaluate our work, we test our models, together with a strong baseline, using untrained automatic metrics, machine-learned metrics and human-centric metrics, allowing to get an hollistic and complete comparison of the different methods employed on our task.

1.3 Organization of the Document

This thesis is organized as follows: Chapter 2 presents the related work, clarifying its utility and how each of the works is related to the present one. Chapter 3 presents the relevant datasets to our work. Chapter 4 describes the methodology of our work and the proposed models. Chapter 5 explains how the work is evaluated. Finally, Chapter 6 presents the main conclusions and possible future work.

[1] https://www.kaggle.com/thoughtvector/customer-support-on-twitter

2 Related Work

Contents

In this Chapter, we start by presenting the available *corpora* related to the context of our problem. Then, we move on to presenting different works previously used to tackle similar problems. This includes works related to Customer Support and Open-Domain dialog systems.

2.1 Dialog Systems

A dialog system, commonly referred to as Conversational Agent or Chatbot, is a computer system intended to converse with a human in Natural Language. Depending on their application, according to [Chen et al., 2017], they can be classified into two different paradigms :

- **Goal-Oriented Systems**: designed to assist users in carrying out specific tasks in concrete domains, such as booking tickets or making reservations.
- **Open-Domain Systems**: also known as chit-chat bots, these systems are designed with the intention of generating coherent and meaningful responses in an open-domain system, maximizing user engagement.

2.2 Sequence Modeling Methods

When developing end-to-end dialog systems, sequence modeling methods are among the most popular. These methods can be divided into recurrence-based, convolution-based, and self-attentional methods.

In recurrence-based methods, a Recurrent Neural Network (RNN) processes each word of the input utterance in a sequential manner (Figure 2.1). At each timestep t, the input x_t and the state vector s_{t-1} are used to compute the hidden state h_t. The RNN can be trained using backpropagation to learn optimizing W and b. In its most simple version, h_t is used directly as the state vector s_t for the next input x_{t+1}. This can be formally defined as:

$$\begin{aligned} h_t &= g(x_t W^x + s_{t-1} W^s + b) \\ s_t &= h_t \end{aligned} \tag{2.1}$$

In equation 2.1, W represents the weight matrices that apply a linear transformation to x_t and s_{t-1}. The variable b represents the bias. The variable g represents a non-linear activation function (such as tanh, sigmoid or Rectified Linear Unit (ReLU)).

The RNN is recurrent in nature since it applies the same function for every input. However, this computation depends on the current input and the hidden state, making it applicable to tasks where the inputs are not independent of each other.

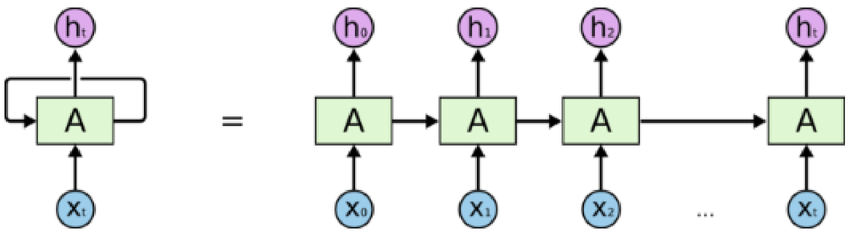


Figure 2.1: Unrollment of a simple recurrent neural network. Taken from: `http://tinyurl.com/q6dcybc`

In some use cases, it is beneficial to not only learn the dependencies of an input from previous inputs but also the dependencies from future inputs. This leads to the creation of the bidirectional RNN [Goodfellow et al., 2016]. The RNN is, in theory, capable of encoding an entire sequence x of inputs in the last hidden state. However, empirically they display a tendency for vanishing gradients. This happens when the gradient values become too small and the model starts learning considerably slower, taking too many iterations to improve. This prevents **RNNs!** to be able to model long-term dependencies in the input sequence. To tackle this problem, [Hochreiter and Schmidhuber, 1997] proposed the Long Short-Term Memory (Long Short-Term Memory (LSTM)) architecture (Figure 2.2). The LSTM is a modified version of the RNN, that was designed to handle long-term dependencies. Like the RNN, it can be trained using back-propagation. The LSTM uses 3 gates:

- **Input Gate**: selects which values from the input will affect the memory. A sigmoid function decides which values are selected and a tanh function is used to weight the importance of the selected values.

$$\begin{aligned} i_t &= \sigma(W_i[h_{t-1}; x_t] + b_i) \\ C_t &= tanh(W_c\dot{[}h_{t-1}, x_t] + bc) \end{aligned} \tag{2.2}$$

- **Forget Gate**: selects which values from the cell state should be forgotten. To make this decision it uses a sigmoid function:

$$f_t = \sigma(W_f[h_{t-1}; x_t] + bf) \tag{2.3}$$

- **Output Gate**: This gate decides the output. Similar to what happens at the Input Gate, a sigmoid function decides which values are selected and a tanh function is used to weight the importance of the selected values.

$$\begin{aligned} o_t &= \sigma(W_o[h_{t-1}; x_t] + bo) \\ h_t &= ht = o_t\dot{t}anh(C_t) \end{aligned} \tag{2.4}$$

Despite its benefits, these models leveraging the RNN architecture still present a significant limi-

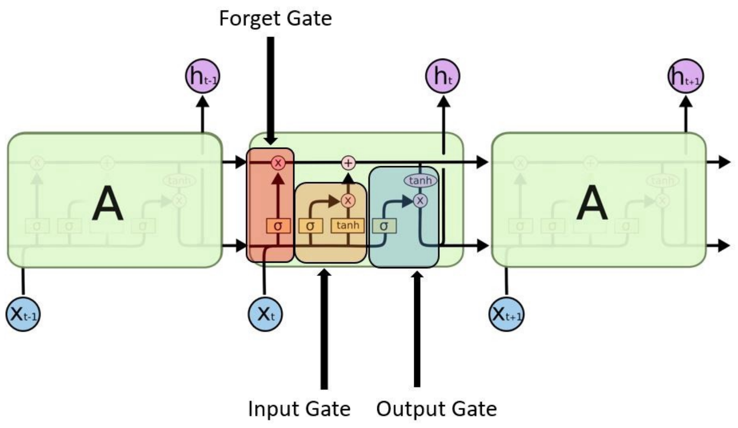


Figure 2.2: Visual representation of the LSTM and its gates. Taken from: `http://tinyurl.com/yy79zmxo`

tation: the output length must be the same as the input length. To deal with this, [Sutskever et al., 2014] proposed the Seq2Seq as a general framework that adapts the RNN (Figure 2.3). This type of architecture has shown great results in multiple different tasks, including machine translation and dialog systems. In a Seq2Seq architecture, the model consists of two **RNNs!**, the encoder, and the decoder. The encoder reads an input and outputs a fixed-length vector, and the decoder reads that vector to produce a variable-length output sequence. Both **RNNs!** are jointly trained using gradient descent to maximize the conditional probability of the output sequence, given the input. This can be expressed as minimizing the cost function:

$$J_\sigma = -\frac{1}{|Y|}\sum_{i=1}^{n} log(P(y_i|x_1, ...x_m, y_1, ..., y_{i-1})) \tag{2.5}$$

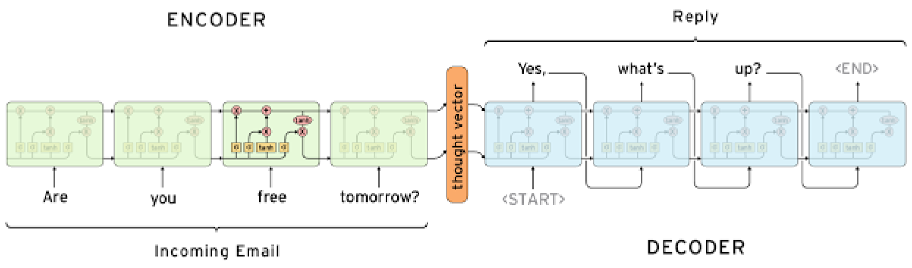


Figure 2.3: Visual representation of the Seq2Seq architecture. Taken from: `https://tinyurl.com/he3neh5`

Besides **RNNs!**, convolutional methods have also been applied. In this case, a Convolutional Neural Network (Convolutional Neural Network (CNN)) architecture is used [LeCun et al., 1998]. Even though this type of model is mostly used for vision tasks, it has been successfully applied to sequential tasks. In this case, the stacking of convolutional and pooling layers allows the model to capture complex dependencies between the elements of the input [Gehring et al., 2017].

Finally, there are also self-attentional methods (Figure 2.4). These rely solely on attention mechanisms. The attention mechanism, proposed in [Bahdanau et al., 2015], serves to allow the decoder to focus on specific parts of the encoder's output at each step. To do this, firstly a set of attention weights are calculated, then these are multiplied by the encoder's output to create a weighted combination. The calculation of attention weights is done with a Feed-Forward Neural Network (Feed-Forward Neural Network (FFNN)) layer that takes the decoder's input and the current hidden state as inputs. The probability of an output is now conditioned on a different context vector c_i for each output word y_i instead of the original single context vector c. Each context vector c_i is obtained by a weighted sum of the encoder hidden states:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j \tag{2.6}$$

where α_{ij} is the attention score or alignment score, representing how relevant h_j to generate the next word of the output. The attention scores α_{ij} are typically obtained through a softmax function and the attention layer is jointly trained with the remaining model.

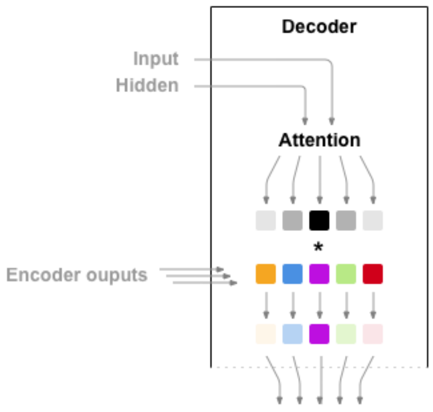


Figure 2.4: Visual representation of the attention mechanism on a Seq2Seq architecture. Taken from: `https://tinyurl.com/yxu89jjf`

Self-attentional methods represent a new paradigm for sequential modeling since their architecture is designed relying solely on self-attention mechanisms for both the encoder and the decoder. The Transformer, proposed in [Vaswani et al., 2017], is possibly the most famous self-attentional model and has been on the core of various models that achieved state-of-the-art results in a myriad of different tasks (Figure 2.5). In the Transformer, the encoding component is a stack of structurally identical encoders and the decoding component is a stack (of the same size) of structurally identical decoders. Each encoder is comprised of a self-attention layer and an FFNN. The decoder is similar, but between these two layers, there is an attention layer that helps the decoder focusing on the most important parts of the input at each step. In the encoder, the self-attention layer allows the Transformer to understand what other parts of the input are relevant for the encoding of each word. To calculate the output of this layer, we must start by creating three vectors for each of the input words: **Query vector**, and **Key vector**, both of dimension d_k, and **Value vector** of dimensions d_v. These are obtained by multiplying the embedding of each word and the weights W^Q, W^K, and W^V respectively. These weights are learned during training. The second step is to calculate the self-attention score. This score is obtained through the dot product of the query vector and the key vector of each word. Then, these scores are divided by the square root of the dimension of the key vector: $\sqrt{d_k}$ and passed through a softmax function that normalizes these scores. This score will represent how much each word is expressed at each position. Finally, we multiply each value vector by the softmax score obtained and sum the results. This produces the output of the self-attention layer for each position. The resulting vector is then passed to the FFNN. The output of each encoder is passed on to the next encoder on the stack until it reaches the last one. In order to account for the position of the words in the input sequence, the Transformer adds a positional embedding to each input embedding. The output of the last encoder is mapped into a set of attention vectors K and V. These are then used by the decoder in its attention layer to focus on the most relevant parts of the input sequence. The output of the encoding stack, at each timestep, is passed on to the first decoder in the next timestep, which will process it and pass it along the stack (similarly to what happened in the encoding phase). This process is repeated until a special symbol is reached indicating the transformer decoder has completed its output. However, a key difference in the decoder is that the self-attention layer can only take into consideration earlier positions in the output sequence, which is done by masking future positions. Finally, the model has a Linear layer and a Softmax layer. These are responsible for converting the final embeddings into actual words to produce the output.

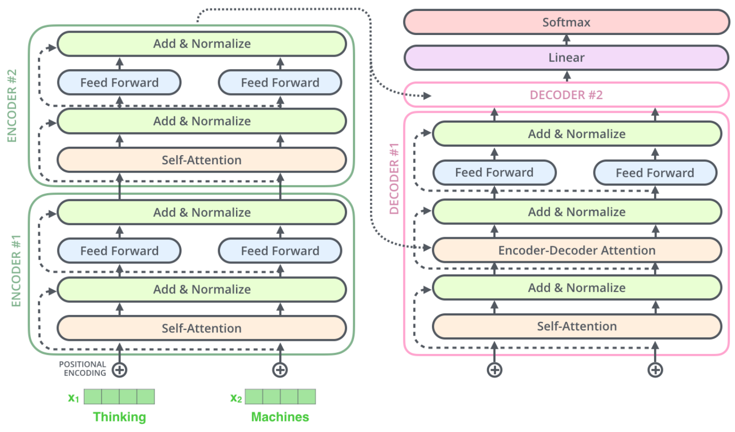


Figure 2.5: Visual representation of a transformer with 2 encoders and 2 decoders. Taken from: https://tinyurl.com/y4gtcmwu

2.3 Persona-Based Dialog Systems

In open-domain, dialog systems have seen incredible advances since the first system of this kind, Eliza [Weizenbaum, 1966], was created. Nowadays, with large volumes of human-to-human interaction data available, a lot of end-to-end trained models have achieved great success in this type of setting. However, a conversation with one of these models quickly reveals some of its shortcomings. Namely, their tendency for vague answers, and, when not vague, often revealing their inability to maintain a coherent persona (as defined in [Li et al., 2016]). Endowing dialog systems with a coherent persona allows us to better model human behavior, leading to more realistic and natural conversations. Furthermore, this problem is also interesting in the context of Customer Support since we can see this persona as the archetype of the Customer Support agent for a brand. Then, by endowing a dialog system with such a persona, we can deliver an agent that better corresponds to the expectations of both the company and the customer. In [Wolf et al., 2019], a new approach to data-driven dialog systems called TransferTransfo is introduced. This new dialog system is based on a multi-layer Transformer architecture [Radford, 2018] (Figure 2.6). It uses a 12-layer decoder-only transformer with masked self-attention heads (768 dimensional states and 12 attention heads). Following the work in [Radford, 2018], the model is pre-trained on the BooksCorpus dataset [Zhu et al., 2015], containing more than 7 000 unpublished books (close to 800M words). This dataset was chosen to take advantage of long contiguous sequences, giving the model the opportunity to learn long-range dependencies. Then, the model was fine-tuned on the Persona-Chat dataset. In this step, the model receives as input a single sequence that is manipulated to include information about the speaker's personality and the context of the conversation. In order to include these elements, these segments are prepended to the corresponding utterance. In order to help the model understand that these segments have different meanings, delimiter tokens and segment embeddings are introduced. The delimiter tokens are used to separate the different segments: speaker personality, context, and utterance. The segment embedding has different values for each of the segments and is added to the word embeddings. The speaker personality and context tokens will be masked, meaning that the model will not learn to predict words from those segments. However, it will facilitate the generation of the reply word by word by taking these segments into consideration. The fine-tuning is done in a multi-task setting, trying to minimize the two distinct losses:

- Language modeling: the final hidden state is passed to a softmax layer, resulting in a probability distribution over the vocabulary.
- Next-sentence prediction: the hidden-state of the last token is passed through a linear layer to get a score and apply a cross-entropy loss to classify correctly a gold answer among distractors. These distractors are utterances randomly selected from the dataset (excluding the gold reply).

The total loss is given by the weighted sum of these two.

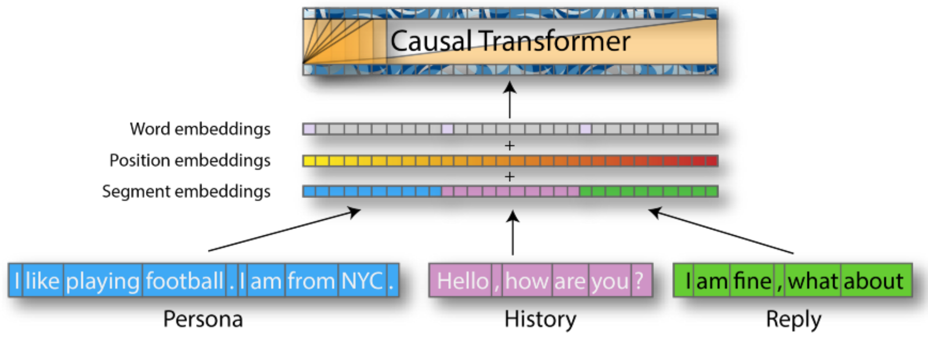


Figure 2.6: Visual representation of the TransferTransfo architecture. Taken from: `https://tinyurl.com/ryehjbt`

This architecture proved to be successful in building a dialog system that is able, to some extent, to enact a persona described in a few persona sentences. Therefore, it might be a great starting point for a brand-aware Customer Support dialog system.

3 Datasets

Contents

Some of the *corpora* presented below are not directly applicable to the problem we are trying to tackle. Notwithstanding, they can be very interesting in terms of developing baselines or pre-training models that can then be fine-tuned for our objective.

3.1 Open WebText Corpus

The Open WebText Corpus [Radford et al., 2019] contains around 40GB of high-quality text from over 8 million documents. To create it, all outbound links from Reddit (an american social platform), that received at least 3 karma, were scraped. This criterion was used as a heuristic indicator for whether the other users found the posted link interesting. All Wikipedia documents were removed from WebText since it is a common data source and that could skew its analysis in evaluation tasks. The resulting corpus includes text from 45 million such links.

3.2 BookCorpus Dataset

The BookCorpus Dataset [Zhu et al., 2015] is a corpus composed by 11 038 unpublished books publically available on the web. All the books in the dataset have more than 20 000 to avoid noisier short narratives. It includes a wide variety of books, including 16 different genres, such as Fantasy, Romance, Science Fiction, and others.

3.3 Twitter Customer Support Dataset

This dataset was developed to “aid innovation in natural language understanding and conversational models, and for the study of modern customer support practices and impact” [1]. With 2 811 774 tweets from customers and Customer Support agents from 20 major brands from different business areas, including Apple, Amazon, Uber, Delta, Spotify, Tesco, and others. This is one of the largest publicly available real Customer Support datasets and therefore, a great starting point to our study.

3.4 MultiWOZ

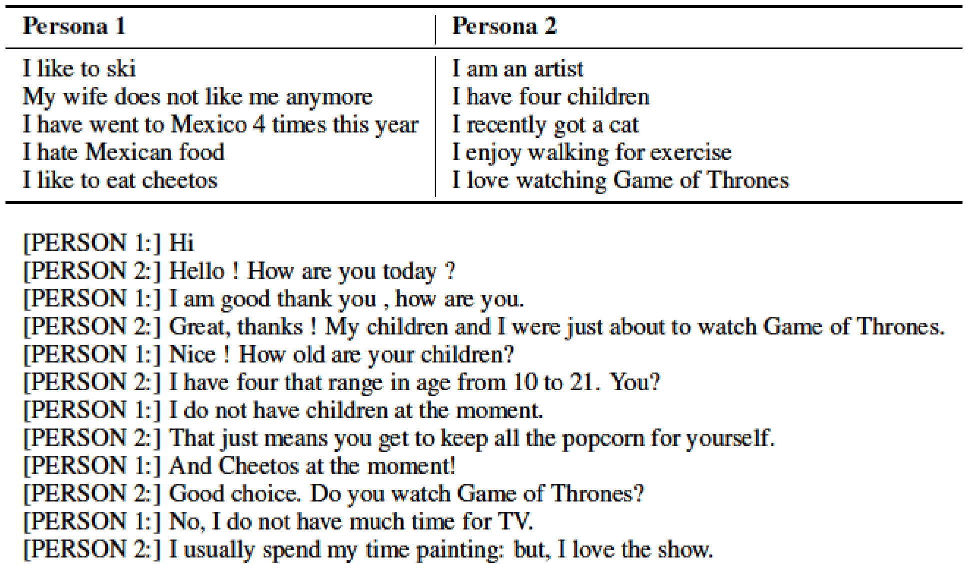
One of the main challenges in the Customer Support setting has to do with data scarcity. To tackle this problem, the MultiDomain Wizard-of-Oz dataset (MultiWOZ) was created [Budzianowski et al., 2018]. This dataset contains a fully-labeled collection of human conversation regarding a myriad of topics. With over 10k dialogs, it is one of the largest annotated task-oriented corpora. Because of its dimension, the

[1] https://www.kaggle.com/thoughtvector/customer-support-on-twitter

corpus can be used for end-to-end modeling experiments, making it extremely relevant for the task at hand.

3.5 Persona-Chat Dataset

The Persona-Chat dataset [Zhang et al., 2018] consists of 162 064 utterances distributed over more than 10 000 dialogs (Figure 3.1). The dataset was created by crowdsourcing "persona sentences" and then, randomly pairing another set of crowd workers. Each crowd worker of the second set was randomly assigned a persona (a small subset of the "persona sentences") and asked to chat naturally while trying to get to know their pair during the conversation. This setup allows the creation of engaging conversations that allow models to learn the relation between the "persona sentences" and the utterances of that persona. Even though the utterances in this dataset are representative of a chit-chat scenario, the idea of imbuing an agent with a persona can be closely related to making the agent brand-aware in the context of Customer Support and, therefore, this can be extremely useful in our work.



Persona 1	Persona 2
I like to ski	I am an artist
My wife does not like me anymore	I have four children
I have went to Mexico 4 times this year	I recently got a cat
I hate Mexican food	I enjoy walking for exercise
I like to eat cheetos	I love watching Game of Thrones

[PERSON 1:] Hi
[PERSON 2:] Hello ! How are you today ?
[PERSON 1:] I am good thank you , how are you.
[PERSON 2:] Great, thanks ! My children and I were just about to watch Game of Thrones.
[PERSON 1:] Nice ! How old are your children?
[PERSON 2:] I have four that range in age from 10 to 21. You?
[PERSON 1:] I do not have children at the moment.
[PERSON 2:] That just means you get to keep all the popcorn for yourself.
[PERSON 1:] And Cheetos at the moment!
[PERSON 2:] Good choice. Do you watch Game of Thrones?
[PERSON 1:] No, I do not have much time for TV.
[PERSON 2:] I usually spend my time painting: but, I love the show.

Figure 3.1: Example of an entry of the Persona-Chat Dataset. Taken from [Zhang et al., 2018]

3.6 Friends Corpus

The Friends Corpus [Zhou and Choi, 2018] is collection of all the conversation occurring over the 10 seasons of Friends (the popular America TV sitcom from the 1990s). The series have a total of 236 episodes, with 3 107 conversations, 67 373 utterance and 700 characters. Even though this *corpus* was created with a different task in mind (Character Identification), the speaker-level and utterance level metadata make it ideal for the development of our model in a Open-Domain environment.

4
Data Tool & Developed Models

Contents

The first step to develop our models was to create ”Piertotum”, a data preprocessing tool that allowed us preprocess our datasets in a quick and programmatic way. Then, following the most recent works presented in Chapter 2, we explored different Transformer-based language models that have successfully been employed to solve tasks in both Open-Domain and Goal-Oriented settings. As a baseline, we developed a dialog system based on the DialoGPT. To improve on its results, we develop a two models still based on DialoGPT. The first one has newly trained embeddings that allow the usage of special tokens in our vocabulary to represent delimiters and segment indicator. This allows us to feed, as input to the model, not only the conversation history, but also some meta-data about the persona. The second one, leverages the use of adapters [Houlsby et al., 2019], a set of newly introduced weights within the layers of the initial model. This provides an alternative method to fine-tune the model to each persona.

4.1 Piertotum

In order to adapt the ”Customer Support on Twitter” dataset and ”Friends Corpus” dataset, we decided to develop a tool that would allow us to do it programmatically. To this end we developed “Piertotum”. The tool includes three main modules:

- GetMetadata;
- Preprocess;
- Personify.

The GetMetadata module allows us to retrieve metadata related to the dataset. This includes number of utterances, number of conversations, number of brands, size of the dataset and others. The “Preprocess” module allows us to preprocess the dataset by applying specific steps that can be controlled through the use of its options:

- emojis: a Boolean that if True, removes all the emojis from the dataset. These can be useful for tasks like sentiment analysis. However, in this scenario, it is better to remove them (since we want to generate text). The default value is True;
- emoticons: a Boolean that if True, removes all the emoticons from the dataset. Similar to emojis, for our goal, it is better to remove them. The default value is True;
- urls: a Boolean that if True, tags urls in the dataset as ’(URL)’. Often, the agent directs the user to an url in order to assist him. Since the specific url is not relevant, in this case, we decided to mask it as the token ’(URL)’. The default value is True;

- html_tags: a Boolean that if True, removes all the html tags from the dataset. The default value is True;
- acronyms: a Boolean that if True, converts acronyms to their meaning. E.g: ”SMH” is converted to ”So Much Hate”. In Twitter, the use of specific abbreviations is extremely common. For the purpose of fine-tuning the model, it may be beneficial to substitute them for their meaning. To do this, we created a list of the most common abbreviations and their meaning that the tool uses to convert. The default value is True;
- spelling: a Boolean that if True, spell checks the dataset. This can be an extremely useful feature since there are a lot of spelling errors present in the dataset. However, since the operation of spell checking all utterances is very time-consuming, we decided to not do it at this stage. The default value is False;
- usernames: a Boolean that if True, tags usernames as ”USER” and ”AGENT” for users and agents, respectively. The default value is False.

Finally, the “Personify” module allows us to format the preprocessed dataset as the Persona-Chat dataset. Instead of a speaker personality, each brand has in this field either its name or a short description of the brand characteristics. It includes the options:

- brand: a String that represents the name of a brand. If not null, the tool will only adapt the conversations with agents representing that specific brand. If the value is null, it adapts the whole dataset. The default value is Null;
- limit: an Integer that indicates how many conversations we want to adapt. This option can be useful in scenarios where we have limited computational resources. If the value is -1, it uses the whole dataset. The default value is -1.

Besides the main modules, the tool includes a list of the most common acronyms used on Twitter, and a dictionary with all the emojis and emoticons with their name, representation, and meaning. This can be useful if, in the future we want to substitute the emojis for their meaning instead of erasing them from the utterance.

In order to interact with the different modules we developed a Command Line Interface (CLI). A CLI is a text-based user interface, that allows us to interact with the scripts through the command line. In this case, our CLI allows us to choose the module we want to use and specify its options. To run it we can write on the terminal:

```
$ python cli.py [module_name] [--option]*
```

The complete tool, with its description and instructions on how to install and run it can be found in public repository on Github [1].

4.2 Developed Models

4.2.1 Baseline

The first model developed and used as a baseline is a multi-layer Transformer decoder based on the work described in [Radford, 2018]. Architecturally, it inherits from GPT-2, a 12 layer decoder-only transformer with 12 masked self-attention heads and 768 dimensional states. As explained in Chapter 2, the masked attention mechanism works a constraint on self-attention, making it so every token can only attend to the tokens on its left (left context).This mechanism is what gives it the name of "Transformer decoder" since it identical to the decoder of the original encoder-decoder Transformer [Vaswani et al., 2017]. Similarly to the models in [Radford, 2018] and [Devlin et al., 2018], the model leverages positional embeddings with a maximum sequence size of 512 tokens. It also preprocesses and tokenizes the input sequences using Byte Pair Encoding (BPE) with Vocabulary size of 40.000 words [Sennrich et al., 2016].

We chose this model as baseline because the text generated by GPT-2 is extremely coherent and it has had recent success in several NLP tasks demonstrates that Transformer Language Models (LMs) are able to portray natural language to a fine level of detail. To bootstrap our work, we based the implementation of our model on the Pytorch adaption of GPT-2 published by Huggingface. [2]

4.2.2 TransferTransfo

Architecturally, this model is identical to the baseline. However, its training, particularly the fine-tuning phase is fundamentally different, as described in chapter 5 the second model is trained following the methodology described in [Wolf et al., 2019], leading to contrasting results. For that reason and for the sake of clarity, we will be treating them as different models.

4.2.3 Adapter

The third and last developed model has the same base as the previous models. However, it has a key architectural difference: the introduction of domain adapters. As described in chapter 2, instead of fine-tuning the model as a whole, the use of adapters allow us to only fine-tune a small set of task-specific parameters. The remaining weights are kept fix. This brings advantages in terms of size, modularity and composability without sacrificing the quality of the results. The applied adapters are based on the work

[1] https://github.com/HLT-MAIA/twcs2PersonaChat
[2] https://github.com/huggingface/pytorch-openai-transformer-lm

described in [Houlsby et al., 2019], where the adapters are applied to NLP by fine-tuning them to learn representations for specific downstream tasks (sentiment analysis, question answering and others). In our work, we use them as persona-adapters, where instead of different downstream tasks, the adapters learn representations for each of the personas in the dataset. The implementation of adapters present multiple architectural choices that allows the developer to control the concrete structure of its modules and their location in the layers of the Transformer. Previous works suggest that simple designs attain good performance, with empirical results on par with more complex counter-parties [Houlsby et al., 2019, Pfeiffer et al., 2020, Stickland and Murray, 2019]. Besides that, our task seems, a priori, simpler than the tasks adapter are usually used for. For these reasons, we implemented a fairly simple adapter design, following [Houlsby et al., 2019, Pfeiffer et al., 2021]: On each Decoder unit, we insert two adapter components, one after the multi-head attention and one after the feed-forward layers. Each of these components is comprised of a two-layer feed-forward neural network with a bottleneck and a nonlinear activation function between the projection layers.

5

Experiments

Contents

5.1 Data Preparation

For the training of our models we use two datasets, one for Customer Support (Goal-Oriented) and one for Open-Domain. In the first setting, the dataset used for fine-tuning the models is based on the “Twitter Customer Support Dataset” (3.3). However, the dataset must first be adapted to our end goal. To do this, the following preprocessing steps are required:

First, we must preprocess each utterance. This includes lowercasing, removing emojis/emoticons, and anonymizing IDs. Besides this, we also must remove all HTML tags and substitute all URLs for an URL token. Finally, we must also convert commonly used acronyms for their meaning (e.g.:“afk” is substituted by “away from keyboard”). We also need to eliminate all non-English utterances. After having the utterances preprocessed, we rebuild the sequence of the conversation between each customer and the respective Customer Support agent. To rebuild these conversations, we start by identifying the tweet that begin a conversation (usually initiated by a customer) and then trace the following tweets in that thread using their IDs. Thanks to the utterances’ preprocessing, some of these conversations will have non-valid utterances (e.g.: utterances consisting only of emojis will become empty). For this reason, before returning the final conversations, we must eliminate all the conversations that have non-valid utterances. In the end, each conversation includes a list of objects composed by:

- an utterance;
- its context with all the prior utterances in that conversation;
- the speaker (customer or agent);
- the brand name or a short description of the brand characteristics.

The final dataset is composed of all the remaining utterances after preprocessing and organized into conversations (Figure 5.1). To build this final dataset, we will leverage the tool we created, “Piertotum” (4.1), which allows the selection of different preprocessing steps and the addition of various metadata to each conversation.

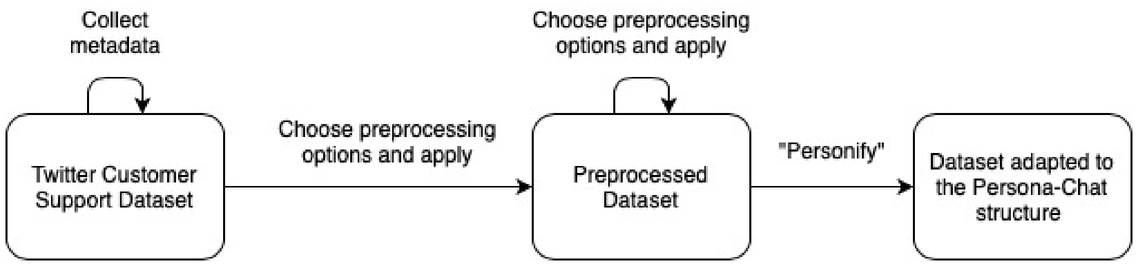

Collect metadata
Choose preprocessing options and apply
Twitter Customer Support Dataset
Choose preprocessing options and apply
Preprocessed Dataset
"Personify"
Dataset adapted to the Persona-Chat structure

Figure 5.1: Visual representation of the preprocessing pipeline.

For the Friends Corpus [Zhou and Choi, 2018], the preprocessing is similar but far simpler. In this dataset we can skip the anonimization step, removal of non-textual elements and expansion of acronyms. The remaining steps are identical.

5.2 Training

Since we want to test the same approaches both in Open-Domain and Goal-Oriented settings, we must train the models twice (one for each of these settings). For each of these settings, the training is identical except for the used dataset. For Open-Domain, we use the “Friends Corpus” and for Goal-Oriented we use the “Twitter Customer Support Dataset”. As such, the methods described in this section were applied twice (once for each dataset).

5.2.1 Baseline

Our baseline, is based on the architecture of DialoGPT, an extension of GPT-2 designed to address the challenges of conversational response generation.

5.2.1.A Pre-Training

Following previous works ([Radford, 2018, Wolf et al., 2019]), the pretraining of the model is done employing the BookCorpus Dataset (described in Section 3.2). The rational behind this choice has to do with the fact that this is a document-level corpus and not a shuffled sentence-level corpus. By exposing the architecture to long contiguous texts, it can better learn how to model long-term dependencies. That would not be the case with a shuffled sentence-level corpus. The model uses a vocabulary of 50 257. A Noam learning rate scheduler with 16000 warm-up steps was used for the pre-training. This corresponds to increasing the learning rate linearly during the first ”warmup steps”, and then, decreasing it thereafter proportionally to the inverse square root of the step number, as described in [Vaswani et al., 2017].

5.2.1.B Fine-Tuning

After having the model pre-trained we now must choose a loss to fine-tune the model on. Following the work of [Wolf et al., 2019], we use a combination of two different losses: language modelling loss, and next-utterance loss.

The first is a simple cross-entropy loss. The softmax function is applied to the content of the last hidden state of the last decoder unit. This will return the next token probabilities. The target values are used as labels and, based on that we calculate the Negative Log Likelihood (NLL).

To calculate the other loss we need add an extra layer to our model. A linear layer is appended after the last decoder. This layer works as a classifier. The classifier is trained to identify the correct next sentence among a group of distractors (these are a group of 2-6 randomly sampled sentences from the training dataset). As an input, the model takes the an element composed by the last hidden state and the next sentence/distractor. Then, a score is calculated for each if these elements and these scores are passed to a softmax function to obtain the probabilities associated with each of them. This classifier is jointly trained with the model fine-tuning.

For the fine-tuning step we use a batch size of 16 for 2 epochs. We used the Adam algorithm [Kingma and Ba, 2015] as an optimizer, with learning rate 6.25e-5, an exponential decay rate for the first moment ($\beta 1$) of 0.9, an exponential decay rate for the second moment ($\beta 2$) of 0.999, and L2 weight decay of 0.01. Next we had to decide what are the relatives weights given to each loss. As in the referenced works, we chose giving a weight of 75% to the language modelling loss and 25% to the cross entropy loss.

A dropout probability of 0.1 was given to all the layers and, following [Radford, 2018] we used ReLU as activation function.

5.2.2 TransferTransfo

The training of the second was identical in all aspects to the training of the baseline. However, a change was made to the input, based on the work described in [Wolf et al., 2019]. During the fine-tuning step on this dataset, the name or a short description of the persona characteristics take the place of the speaker personality on the original model. In the Open-domain setting this means the name of the character or a description of their personality. In the Goal-Oriented domain this means the name of the brand or a description of its characteristics. We experiment with these two variations since each one of them may present different benefits. On one side, if the persona's name coupled with the interactions is enough for the model to learn how to impersonate it, adding new personas to our dialog system becomes trivial, provided we had enough data. On the other hand, if providing a short description of the persona can be used for the same end, the model may present a higher generalization capability, since it may be able to simulate the communication style of personas that were not in the dataset, provided that they can be described as a combination of the sentences used to describe personas that were.

Then, these persona sentences are used to generate an augmented input representation. An input is generated by concatenating the persona sentence(s), the conversation history (clipped to prevent the input to become too large). Additionally, extra tokens are created to separate the sequences. This input is then used to generate the initial embeddings the model will consume. Besides the original word and positional embeddings learned in the pretraining, a third set of segment embedding are used to indicate to which segment of the input does each token belong to. This set of embeddings is trained during the fine-tuning phase. The sequence passed to the initial decoder block is the sum of the three (word, positional and segment) embedding arrays.

5.2.3 Adapters

Once again, the pre-training of this model is the same as the previous ones. However, the fine-tuning phase is fundamentally different. Instead of fine-tuning all the weights of the Transformer, only a small subset of the weights are updated during the fine-tuning (the ones belonging to the adapter modules), while the rest are kept fixed.

The initial Adapter weights are set with a near identity initialization, which is required for a stable training of the model [Houlsby et al., 2019]. The training is done following the approach in [Devlin et al., 2018]. To this end we use an initial learning rate of 0.0001 and optimize it with Adam algorithm [Kingma and Ba, 2015]. We try them with reduction factors 16, 64.

5.3 Evaluation Metrics

According to [Celikyilmaz et al., 2020], Dialog System evaluation methods can be grouped into three categories: untrained automatic metrics, machine-learned metrics, and human-centered evaluation metrics. Since Dialog Generation is, for the most part, an open-ended problem, the evaluation of Dialog Systems can be extremely challenging. Many of the existing automated metrics were adopted from Machine Translation and have been shown to be sub-optimal for the evaluation of Dialog Systems [Liu et al., 2016a]. For this reason, human evaluation is still regarded as the gold standard for this type of task. However, human evaluation can be considerably expensive and time-consuming, making it impractical for quantifying day-to-day progress or for dealing with performance optimization matters.

With this in mind, we will use different metrics for different stages of development, leveraging the three categories of evaluation methods, allowing to measure the models' quality from different perspectives and in the most efficient manner.

5.3.1 Untrained Automatic Metrics

This is the most commonly used type of metric. Untrained Automatic Metrics evaluate the system by comparing the machine-generated texts to human-generated texts, for the same input. The difference when compared to machine-learned metrics, is that the former compares these texts using simple rules, such as n-gram matching or distribution similarity, making it considerably faster and less expensive to calculate. For this reason this type of metrics is ideal to use in day-to-day development and for performance optimization.

Following the proposed evaluation in [Budzianowski and Vulic, 2019], we consider Bilingual Evaluation Understudy (BLEU) [Papineni et al., 2002] as the main metric to measure the fluency of the generated answers. This metric analyses the overlapping of n-grams between the machine-generated answer and a set references. Even though, it was developed with the task of Machine Translation in mind, it is commonly used in other NLP tasks, particularly in dialog generation. sentences, regardless of the word order.

5.3.2 Machine-Learned Metrics

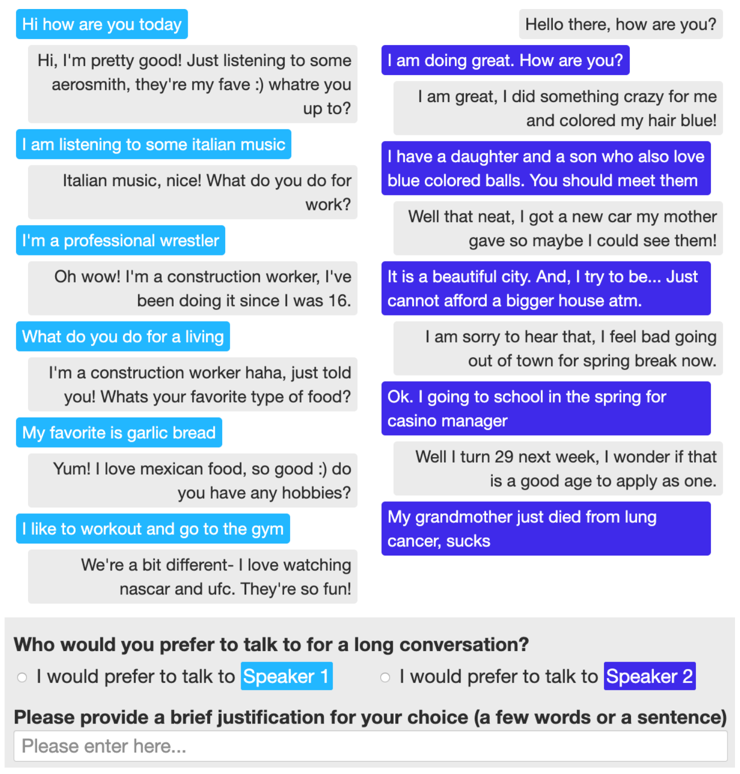
These metrics usually leverage machine-learning models that will measure the semantic similarity between texts. These models try to simulate a human judge, offering a cheaper alternative to Human-Centered Evaluation. One of the most famous machine-learned metrics is the BERTScore [Zhang et al., 2019a]. Like BLEU, BERTScore calculates the similarity between the tokens of two sentences (the machine-generated and the reference). However, this similarity is calculated using the sum of cosine similarities between the word embeddings for the elements of each sentence. This makes it more context-aware and able to compare the sentences semantically, making more effective, especially in the presence of paraphrases [Devlin et al., 2018].

5.3.3 Human-Centered Evaluation Metrics

These metrics present the most reliable way to evaluate the quality of machine-generated text. Typically, human judges are asked to compare the texts generated by different systems, using single-turn pairwise evaluation or multi-turn Likert scores or to distinguish machine-generated texts using from human-generated texts (Turing Test [Turing, 1950]). However these human judgement tests present some serious flaws. The Acute-Eval method [Li et al., 2019], proposes a novel procedure that involves comparing two full dialogs (Figure 5.2). The human judge is asked to focus on only one speaker within each of the dialogs and then make a pairwise judgement. Besides maximizing the robustness of judgement across different humman judges, using this method also result in faster and cheaper human tests.

5.4 Results

With the evaluation metrics described in the previous section in mind., we proceeded to evaluate the model. First, we compared the three models and their variations using automatic metrics, in both datasets. Then, after selecting the best performing iterations of each model, we performed Human-Centered evaluation for both datasets as well.



Hi how are you today

Hi, I'm pretty good! Just listening to some aerosmith, they're my fave :) whatre you up to?

I am listening to some italian music

Italian music, nice! What do you do for work?

I'm a professional wrestler

Oh wow! I'm a construction worker, I've been doing it since I was 16.

What do you do for a living

I'm a construction worker haha, just told you! Whats your favorite type of food?

My favorite is garlic bread

Yum! I love mexican food, so good :) do you have any hobbies?

I like to workout and go to the gym

We're a bit different- I love watching nascar and ufc. They're so fun!

Hello there, how are you?

I am doing great. How are you?

I am great, I did something crazy for me and colored my hair blue!

I have a daughter and a son who also love blue colored balls. You should meet them

Well that neat, I got a new car my mother gave so maybe I could see them!

It is a beautiful city. And, I try to be... Just cannot afford a bigger house atm.

I am sorry to hear that, I feel bad going out of town for spring break now.

Ok. I going to school in the spring for casino manager

Well I turn 29 next week, I wonder if that is a good age to apply as one.

My grandmother just died from lung cancer, sucks

Who would you prefer to talk to for a long conversation?

○ I would prefer to talk to Speaker 1 ○ I would prefer to talk to Speaker 2

Please provide a brief justification for your choice (a few words or a sentence)

Please enter here...

Figure 5.2: Example of how the dialogs are presented in Acute-Eval. Taken from [Li et al., 2019]

5.4.1 Automatic Evaluation

Tables 5.1 and 5.2 present the obtained results, for the "Friends Corpus" and "Twitter Customer Support Dataset" respectively. Since this is an answer generation task, the model's answer is compared with the ground truth (provided in the dataset) both in terms of word overlapping (for BLEU and METEOR) and embeddings distance (for BERTscore).

	BLEU	METEOR	BERTscore	Hits@1
DialoGPT_m (baseline)	16.34	8.73	75.32	73.98
TTname (greedy)	18.02	9.51	79.21	78.83
TTname (beam)	20.38	10.02	82.07	81.75
TTname (top-k)	20.95	9.23	81.23	79.44
TTsent (greedy)	19.21	9.69	78.56	80.87
TTsent (beam)	22.07	**11.44**	82.41	82.27
TTsent (top-k)	**22.19**	10.74	**83.99**	**82.90**
Adapter (64)	20.44	9.93	77.01	79.83
Adapters (16)	21.48	10.62	82.19	81.70

Table 5.1: Evaluation results for the "Friends Corpus". The models named TTname and TTsent represent the second model based on TransferTransfo with the different inputs (just name of the persona or persona sentences) and different decoding strategies (greedy decoding, beam search, and top-k). The Adapters represent the model using adapters with different reduction factors (64, 16)

	BLEU	METEOR	BERTscore	Hits@1
DialoGPT_m (baseline)	18.14	8.43	78.92	80.01
TTname (greedy)	18.73	9.54	79.61	81.28
TTname (beam)	22.78	10.99	84.25	82.96
TTname (top-k)	22.85	9.33	84.70	79.03
TTsent (greedy)	20.21	10.19	80.56	82.87
TTsent (beam)	**23.74**	11.35	**87.28**	**84.94**
TTsent (top-k)	22.50	9.42	83.24	81.89
Adapter (64)	20.55	10.90	77.09	78.56
Adapters (16)	23.14	**11.04**	86.46	79.88

Table 5.2: Evaluation results for the ”Twitter Customer Support Dataset”. The models named TTname and TTsent represent the second model based on TransferTransfo with the different inputs (just name of the persona or persona sentences) and different decoding strategies (greedy decoding, beam search, and top-k). The Adapters represent the model using adapters with different reduction factors (64, 16)

By analyzing tables 5.1 and 5.2 the first observation that becomes evident is that, overall, the models obtain a significant better performance in the ”Twitter Customer Support Dataset” than in the ”Friends Dataset”. This is likely related to the fact that the first is composed by conversations in a closed domain and the second by conversations in open-domain. In closed domain, the conversations will be more similar, and for the same questions the expected answer will mostly be the same. This is not true for conversations in open-domain, where the expected answer will be more dependent on the context of the conversation, hence the observed difference.

Another interesting observation is that the ”TransferTransfo” is the best performing model for both datasets, however the best decoding strategy is different for each case. Beam-search maintains a beam of the multiple sequences that we can use to construct the answer, word by word. At the end, we select the most likely sequence among the different beams. This is the standard decoding algorithm for most language generation tasks [Kulikov et al., 2018]. However, recent works have shown that beam-search is very sensitive to the length output, and furthermore, that it works best when the output length can be predicted a priori [Murray and Chiang, 2018, Yang et al., 2018]. For that reason, it makes sense that beam-search is a good decoding strategy in low entropy settings (like closed domain dialog generation). However, in higher entropy settings, like Open-Domain dialog generation, where various outputs with different lengths are equally valid for the same input, its performance decreases. For this reason, in open-domain dialog generation, the decoding performance can be improved by leveraging sampling techniques. In this case, we used top-k sampling, in which the model samples the next token from the top-k most likely tokens, being k an hyperparameter. Similarly to previous works [Fan et al., 2018, Radford et al., 2019], in our experiments, this has provided better results than beam searching.

A final observation to be made is that the best performing model is the ”TransferTransfo” model, using persona sentences in the input. As expected, using the persona sentences leads to a more enriched input than using simply the name (since it includes the name and more meta-information, like area of business, etc).

	Trained params (%)	BLEU	METEOR	BERTscore	Hits@1
TTsent (top-k)	100	22.19	10.74	83.99	82.90
Adapters (16)	3.6	21.48	10.62	82.19	81.70
Delta (%)	96.4	3.20	1.12	2.14	1.45

Table 5.3: Comparison between the best overall model and best adapter model for the ”Friends Corpus”. Delta represents the percentage difference of performance in the different evaluated metrics

	Trained params (%)	BLEU	METEOR	BERTscore	Hits@1
TTsent (beam)	100	23.74	11.35	87.28	84.94
Adapters (16)	3.6	23.14	11.04	86.46	82.80
Delta (%)	96.4	2.52	2.73	0.94	2.52

Table 5.4: Comparison between the best overall model and best adapter model for the ”Twitter Customer Support Dataset”. Delta represents the percentage difference of performance in the different evaluated metrics

Tables 5.3 and 5.4 provides a different insight. As expected, fine-tuning the full model provided the best performance in both datasets. However, the use of adapters demonstrate their usefulness. By training only 3.6% of the parameters (instead of fine-tuning the whole model), we can get a performance that is very close to the best developed model.

5.4.2 Human-Centered Evaluation

The usage of automatic metrics allowed us to make an initial evaluation of the models and decide on different parameters. However, this evaluation can be insufficient. Previous works have shown that these metrics do not have a clear correlation with human evaluation [Liu et al., 2016b, Lowe et al., 2017, Peng et al., 2017]. On one side, semantically different phrases can be very similar, and on the other, for the same prompt, multiple different answers may be correct, which makes the automatic evaluation much more difficult, particularly in an Open Domain. For these reasons, after selecting the best models with the automated metrics presented, we evaluated them using a Human-Centered method. The selected method was the Acute-Eval method [Li et al., 2019], ”a novel procedure involving comparing two full dialogues, where a human judge is asked to pay attention to only one speaker within each, and make a pairwise judgment”. The setup is as follows: In each trial, the judge is presented with two previously obtained conversations, one of model A interacting with a human, and one of model B also interacting with the same human. The judge, reads both conversations and is then posed with a question (e.g. ”Which speaker represents Ross from the show Friends”, ”Which speaker represents a Customer Support Agent from Apple”). The judge must choose between model A and B to answer the question. We use 20 annotators, each presented with 5 trials for each pair of models and then use their answers to decide which model wins. The results can be consulted in the tables 5.5 and 5.6.

	Wins (%)			
	Original	DialoGPT	TransferTransfo	Adapter
Original	-	93	88	90
DialoGPT	7	-	24	35
TransferTransfo	12	76	-	61
Adapter	10	65	39	-

Table 5.5: Results of Acute-Eval for the question ”Which speaker do you think represents Ross from Friends?”. The considered models are the baseline (DialoGPT), the TransferTransfo model with beam decoding (TransferTransfo) and the Adapter model with reduction factor of 16 (Adapter). The values for ”Original” are sampled conversations taken from the original dataset.

	Wins (%)			
	Original	DialoGPT	TransferTransfo	Adapter
Original	-	91	85	90
DialoGPT	9	-	46	49
TransferTransfo	15	54	-	58
Adapter	10	51	42	-

Table 5.6: Results of Acute-Eval for the question ”Which speaker do you think represents a Customer Support Agent from Apple?”. The considered models are the baseline (DialoGPT), the TransferTransfo model with beam decoding (TransferTransfo) and the Adapter model with reduction factor of 16 (Adapter). The values for ”Original” are sampled conversations taken from the original dataset.

Similarly to the results obtained with the automatic evaluation, the TransferTransfo model is the one with best performance overall, followed by the Adapter and then the baseline. Another aspect to notice is the fact that, even though all the models are still very far from being indistinguishable from the original conversations, both developed models show a significantly better performance when tested against the original conversations than the baseline. One final observation worth making is that the models are less distinguishable in the Customer Support setting than in the Open Domain. This is due to the fact that the conversations for Customer Support are much more similar and objective, making the conversations with the different models harder to distinguish.

6 Conclusion

Contents

The goal of this work was leverage the state of the art dialog systems that exist today and explore how the can be adapted to represent personas in a scalable way. Solving this problem is not only crucial to deliver a better customer support experience to clients but is also a crucial step to develop more engaging and realistic chatbots, bringing us one step closer to pass the Turing test. This chapter concludes this dissertation by presenting our main contributions in Section 6.1 and pointing out promising directions to further develop this work in Section 6.2.

6.1 Contributions

The main contributions of our work are: (1) the development of *Piertotum*, a preprocessing tool that allows to us to implement a data preprocessing pipeline to NLP datasets and modify their structure; (2) the development of two models capable of incorporating a persona and weaving its traits into their generated answers; an extensive test and comparative analysis of this models and a competitive baseline. *Piertotum*, the preprocessing tool can be used to apply different preprocessing steps to NLP datasets, allowing users to test different preprocessing steps to the datasets and test them. Besides that, *Piertotum* also allows the user to adapt the structure of a dataset to resemble the [Zhang et al., 2018], making it easier to apply the TransferTransfo [Wolf et al., 2019] methodology to any dataset. With the developed models we showed how the TransferTransfo methodology [Wolf et al., 2019] can be leveraged to help Transformer architecture impersonate a predefined persona in Open-Domain and Customer Support settings. The experiments done with this model also allowed us to optimize the use of this method, including the decoding strategy and how to best use the persona sentences. The other developed model, leveraging Adapters [Houlsby et al., 2019], uses this state of the art fine tuning technique in an original way, creating *persona adapters*. The test results show that these *persona adapter* have a slightly lower performance than the other models, with only a fraction of the trainable parameters, making it a competitive and much more scalable approach.

6.2 Future Work

As future work, we would like to further explore *persona adapters*, utilizing the most recent developments in this area, such as AdapterFusion [Pfeiffer et al., 2021] and Mad-X [Pfeiffer et al., 2020]. Due to hardware limitations it was not possible to use the large version of the models to our development, resulting in performance loss. It would also be interesting to test our models using the large version of DialoGPT [Zhang et al., 2019b]. Finally, we would like to explore the usage of GPT-3 [Brown et al., 2020] to solve the posed problem. The reason for this is that, currently, GPT-3 outperforms all other models when it comes to performing of specific tasks without any fine tuning, making it a prime candidate.

Bibliography

[Bahdanau et al., 2015] Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners.

[Budzianowski and Vulic, 2019] Budzianowski, P. and Vulic, I. (2019). Hello, it's GPT-2 - how can I help you? towards the use of pretrained language models for task-oriented dialogue systems. *CoRR*, abs/1907.05774.

[Budzianowski et al., 2018] Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., and Gašić, M. (2018). MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

[Celikyilmaz et al., 2020] Celikyilmaz, A., Clark, E., and Gao, J. (2020). Evaluation of text generation: A survey.

[Chen et al., 2017] Chen, H., Liu, X., Yin, D., and Tang, J. (2017). A survey on dialogue systems: Recent advances and new frontiers. *SIGKDD Explor. Newsl.*, 19(2):25–35.

[Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

[Fan et al., 2018] Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical neural story generation.

[Gehring et al., 2017] Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *CoRR*, abs/1705.03122.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. Book in preparation for MIT Press.

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

[Houlsby et al., 2019] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *ICML*.

[Kaplan et al., 2020] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Kulikov et al., 2018] Kulikov, I., Miller, A. H., Cho, K., and Weston, J. (2018). Importance of a search strategy in neural dialogue modelling. *CoRR*, abs/1811.00907.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition.

[Li et al., 2016] Li, J., Galley, M., Brockett, C., Spithourakis, G., Gao, J., and Dolan, B. (2016). A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 994–1003, Berlin, Germany. Association for Computational Linguistics.

[Li et al., 2019] Li, M., Weston, J., and Roller, S. (2019). ACUTE-EVAL: improved dialogue evaluation with optimized questions and multi-turn comparisons. *CoRR*, abs/1909.03087.

[Liu et al., 2016a] Liu, C., Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L., and Pineau, J. (2016a). How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *CoRR*, abs/1603.08023.

[Liu et al., 2016b] Liu, C.-W., Lowe, R., Serban, I., Noseworthy, M., Charlin, L., and Pineau, J. (2016b). How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132, Austin, Texas. Association for Computational Linguistics.

[Lowe et al., 2017] Lowe, R., Noseworthy, M., Serban, I., Angelard-Gontier, N., Bengio, Y., and Pineau, J. (2017). Towards an automatic turing test: Learning to evaluate dialogue responses. *ArXiv*, abs/1708.07149.

[Lowe et al., 2015] Lowe, R., Pow, N., Serban, I., and Pineau, J. (2015). The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *CoRR*, abs/1506.08909.

[Murray and Chiang, 2018] Murray, K. and Chiang, D. (2018). Correcting length bias in neural machine translation.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

[Peng et al., 2017] Peng, B., Li, X., Li, L., Gao, J., Celikyilmaz, A., Lee, S., and Wong, K.-F. (2017). Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

[Pfeiffer et al., 2021] Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., and Gurevych, I. (2021). AdapterFusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503, Online. Association for Computational Linguistics.

[Pfeiffer et al., 2020] Pfeiffer, J., Vuli'c, I., Gurevych, I., and Ruder, S. (2020). Mad-x: An adapter-based framework for multi-task cross-lingual transfer. In *EMNLP*.

[Radford, 2018] Radford, A. (2018). Improving language understanding by generative pre-training.

[Radford et al., 2019] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

[Sennrich et al., 2016] Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

[Serban et al., 2015] Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2015). Hierarchical neural network generative models for movie dialogues. *CoRR*, abs/1507.04808.

[Stickland and Murray, 2019] Stickland, A. C. and Murray, I. (2019). BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

[Turing, 1950] Turing, A. M. (1950). I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX(236):433–460.

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

[Wallace, 2009] Wallace, R. S. (2009). *The Anatomy of A.L.I.C.E.*, pages 181–210. Springer Netherlands, Dordrecht.

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

[Wolf et al., 2019] Wolf, T., Sanh, V., Chaumond, J., and Delangue, C. (2019). Transfertransfo: A transfer learning approach for neural network based conversational agents. *CoRR*, abs/1901.08149.

[Yang et al., 2018] Yang, Y., Huang, L., and Ma, M. (2018). Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation.

[Zhang et al., 2018] Zhang, S., Dinan, E., Urbanek, J., Szlam, A., Kiela, D., and Weston, J. (2018). Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

[Zhang et al., 2019a] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019a). Bertscore: Evaluating text generation with BERT. *CoRR*, abs/1904.09675.

[Zhang et al., 2019b] Zhang, Y., Sun, S., Galley, M., Chen, Y., Brockett, C., Gao, X., Gao, J., Liu, J., and Dolan, B. (2019b). Dialogpt: Large-scale generative pre-training for conversational response generation. *CoRR*, abs/1911.00536.

[Zhou and Choi, 2018] Zhou, E. and Choi, J. D. (2018). They exist! introducing plural mentions to coreference resolution and entity linking. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 24–34, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

[Zhu et al., 2015] Zhu, Y., Kiros, R., Zemel, R. S., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR*, abs/1506.06724.