Faculty of Human Sciences - Department Linguistics

Investigating and Improving Background Context Consistency in Neural Conversation Models

Dissertation to obtain the academic degree

Dr. Phil.

at the University of Potsdam.

Fabian Galetzka, M. Sc.

2022

Published online on the
Publication Server of the University of Potsdam:
https://doi.org/10.25932/publishup-58463
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-584637

Declaration of Authorship

Hereby, I declare that I have composed the presented paper independently on my own and without any other resources than the ones indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such. This work has only been submitted to Potsdam University and has not been submitted to another degree or professional qualification.

(Fabian Galetzka)

Contents

1. Introduction

Conversation between humans can be seen as a complex task that requires consecutive actions from at least two participants with the general purpose of exchanging information (Wittkenstein, 1953). According to Searle (1969) these actions or *speech acts* can contain directives (such as asking a participant to do something), asserting ideas (e.g., having a suggestion), stating something (e.g., about an attitude towards someone), committing to something in the future (e.g., planning) or declaring something that changes the world (e.g., firing someone from his or her job). These acts are dependent on each other and have a structure: a question, for example, should be followed by an answer. One long-term research goal is to build a machine or model that can maintain such a conversation in all its complexity over any number of exchanges appropriately.

A machine that tries to fulfil the role of a dialogue partner is called a conversational system or spoken (or text-based) dialogue system. These systems can be divided into two major approaches: task-oriented and non-goal-driven systems. The former are usually used as Human-Machine Interfaces (HMI) and are a common part of daily life, such as Amazon's Alexa[1] or Apple's Siri.[2] Their main purpose and strength are to understand and execute *directives* from their users through pre-defined conversations with very few turns (e.g., "What is the weather in Berlin tomorrow?", "It will be sunny the whole day."). Non-goal-driven systems, on the other hand, are built to handle the full complexity of conversations described above. Their main purpose is mostly entertainment, and the type of dialogue is often referred to as *small talk*.

In general, task-oriented dialogue systems operate with a pipeline as follows: First, they transcribe speech into text with an Automatic Speech Recognition (ASR) component, and then estimate a discrete intent and entity values with a Natural Language Understanding (NLU) unit. A dialogue management component handles the dialogue state and, based on a dialogue policy, a Natural Language Generation (NLG) unit produces the returning text that

[1] `https://developer.amazon.com/alexa`
[2] `https://www.apple.com/de/siri/`

has to be transformed back into speech using a Text to Speech (TTS) component (Williams et al., 2016). Non-goal-driven systems are either rule-based and follow similar architectures like the first and important ELIZA model from Weizenbaum (1966) or, more recently, data-driven, which is motivated by the fact that conversation complexity cannot be handled by rule-based systems. Instead, a particular strand of this research was proposed by Ritter et al. (2011) and focus on learning an appropriate dialogue contribution by training neural models with dialogue data sources.

In this thesis, we only consider the non-goal driven systems based on deep neural networks. Over the years, many improvements were made by introducing new and larger data sources and more advanced models. The invention of Self-Attention-based models (Vaswani et al., 2017) also had a significant impact, as many experiments across this field empirically showed that they outperform the widely used Recurrent Neural Network (RNN) models, not only in various automated metrics like perplexity and word overlap, but in a variety of human evaluation experiments as well. In early stages, researchers like Ritter et al. (2011) tried to built models that can capture the structural relationship between an utterance and their immediate reply. Then, expanding this to full variable-length dialogues (Serban et al., 2016) and, more recently, having context-grounded conversations (Zhang et al., 2018; Dinan et al., 2019c). This trend can be described as a transition from just trying to somehow carry on a conversation to generating *appropriate* replies in a controlled and reliable way; which leaves the question open as to what exactly an *appropriate* contribution is.

Grice (1975) proposed the *Cooperative Principle* as a fundamental feature of contributing successfully to a conversation. It consists of nine conversational maxims, which are sorted in four categories: manner, quantity, quality and relation. In this thesis, we first try to define what an *appropriate* contribution has to be by operationalizing these maxims as demands on conversation models and identify different targets (or *intervention points*) to achieve the conversational appropriateness by investigating recent research in that field in Chapter 3. Further, we propose that these demands are being fluent, informative, consistent towards given context, coherent and following a social norm.

While every demand is important, in this thesis we investigate further into consistency towards context. As we will show in Chapter 3, models are already quite fluent (according to our definition) and high informativity is achieved indirectly through manipulating decoding strategies. Securing that models follow a social norm beyond preventing harmful phrases to occur is challenging and in an early research phase. Consistency towards context has got attention recently: context-augmented dialogue datasets that either focus on knowledge (Dinan et al., 2019c) or personality context (Zhang et al., 2018) enabled that brand of research, and many new model ideas emerged that incorporate context into the learning process (Zhao et al., 2020; Lin et al., 2020; Komeili et al., 2021; Fan et al., 2021). Being consistent might also influence informativity and coherence positively, since available knowledge increases the possible content that a model can generate, and a consistent mention of facts, for example, can make a conversation more coherent.

Research towards consistency is mostly divided into context based on facts or context based on personality. We propose a crowdsourced dataset that combines both by forcing participants to talk about movies while following predetermined opinions and providing facts about them. The Knowledgable and Opinionated MOvie DIScussions (KOMODIS) dataset is publicly available [3] and we use the data in this thesis to gather new insights in context-augmented dialogue generation. In Chapter 4 we validate our dataset and conduct experiments on neural conversation models to analyze the generalization capacity for fact- and opinion-based context. For that, we explore different encodings that transform the context into sequences that are added to the dialogue history.

The amount of data (number of token embeddings) that can be fed into a Transformer-based neural conversation model is limited due to how the architecture is designed. Therefore, many approaches try to determine a subset of the available context information at inference based on the dialogue history. In Chapter 5 we propose a concise encoding strategy for structured context inspired by graph neural networks (Gilmer et al., 2017), to reduce the space complexity of the additional context. Our approach greatly reduces the required sequence length for any kinds of graph-based context without losing knowledge precision at all.

The usual language modelling loss approach for training causes problems at inference through error accumulation while decoding token by token. Reinforcement learning enables the use of reward functions that are more suitable for inference, but suffers from the data sparseness in language generation (Mesgar et al., 2021). We investigate the potential of reinforcement learning specifically for consistency in Chapter 6 using a self trained natural language inference model to determine a consistency reward, and show how different reward functions interfere with each other, when trained simultaneously. Finally, our KOMODIS dataset is domain-specific and models fine-tuned on this data tend to forget out-of-domain dialogue behavior learned during pre-training, which is a common issue with neural networks. In the final Chapter 7 we investigate transfer learning strategies with various dialogue datasets to improve the dialogue quality and consistency towards background context, using additional dialogue datasets.

[3] `https://github.com/fabiangal/komodis-dataset`

2. Fundamentals: Transformers for Language Modelling

In this chapter, we clarify the fundamental models, techniques and data sources that are required to understand our experiments and research in the upcoming chapters.

2.1. Transformer Models for Dialogue Generation

Neural conversation model architectures need to process time variant input sequences (of word tokens) and be able to generate a new time variant sequence, which is the continuation of the former. A promising model architecture called *Transformer* was introduced by Vaswani et al. (2017) and resulted in significant improvements across many natural language processing tasks by using their pre-trained models or different variations of it. In this section, we denote and explain the relevant notations of the different architecture types in Subsection 2.1.1 and the fundamental ideas of Self-Attention, their encodings and Future Masking in Subsection 2.1.2.

2.1.1. Architectures

The original *Transformer* has an encoder-decoder architecture. The encoder consists of 6 Self-Attention layers (see next subsection) and transforms an input sequence into a latent representation. The decoder also consists of 6 Self-Attention layers, but each layer has an additional sub-layer that attends the hidden representation from the encoder. One embedding matrix is used to convert the input and output tokens to the learned word embeddings. A linear transformation with a soft-max activation function is used to convert the decoder output to a next-token probability distribution. The model was originally planned for machine translation and is not pre-trained on a specific dataset.

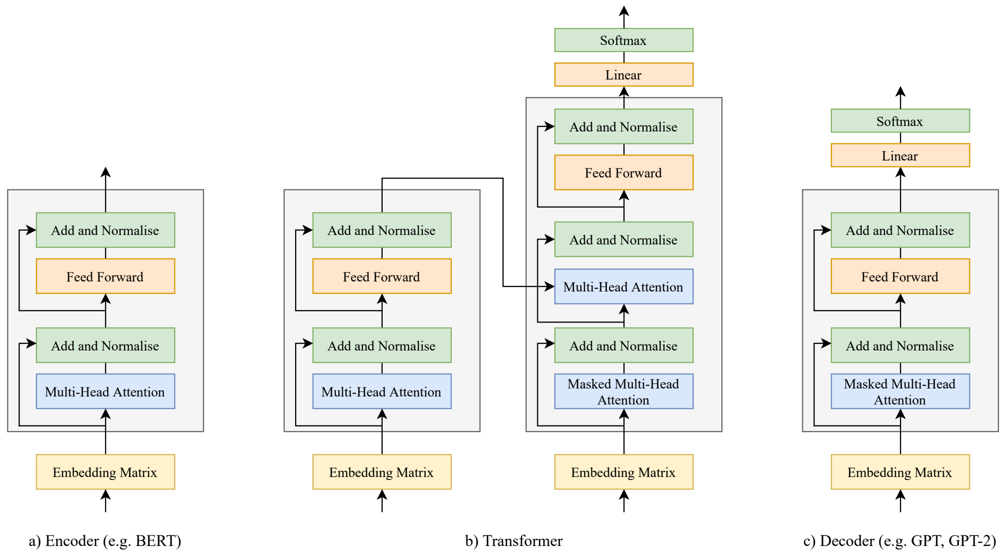

Figure 2.1.: Different architecture variations of the original Transformer (Vaswani et al., 2017), here simplified. Models additionally differ in terms of parametrisation, pre-training and heads (output layers).

From this architecture many new approaches and variations were proposed: The important Bidirectional Encoder Representations from Transformers (BERT) model by Devlin et al. (2019) which is a pre-trained version of the encoder part of the Transformer architecture with different parametrization. And the Generative Pre-trained Transformer (GPT) by Radford et al. (2018) with the evolutions GPT-2 (Radford et al., 2019) and GPT-3 Brown et al. (2020) created for various natural language generation tasks. We visualize the BERT and GPT architectures in Figure 2.1 in comparison to the base Transformer.

BERT The Transformer variation BERT is a natural language encoder model fine-tuned on two unsupervised tasks: masked language modelling and next sentence prediction on the BooksCorpus (Zhu et al., 2015) with 800M words and 2,500M words extracted from the English Wikipedia. In the first task, the model has to predict tokens from the input sequence that are randomly masked out during training with a specific *mask* token. In the second task, the model has to choose the right next sentence for the current input by providing the correct sentence and a randomly sampled sentence from the datasets. This was done by using a specific *classifier* token that is fed into a feed forward layer with a soft-max activation function as a classification head. BERT is available in different sizes with *base* being an architecture with 12 layers, 768 dimensional word embeddings and a total of 110M trainable parameters and *large* being an architecture with 24 layers, 2024 dimensional word embeddings and a total of 340M trainable parameters.

GPT This transformer variation is similar to BERT but uses masked self-attention so that every hidden token representation can only attend to tokens to its left. We will clarify this future masking approach in the next subsection. The GPT model is pre-trained unsupervised on 7,000 unique unpublished books on a standard language modelling objective (explained in Subsection 2.2.1). In that step, the model has to predict the next token given the input sequence. In this thesis, we mostly use the GPT-2 model which is pre-trained on a large corpus generated by scraping the Reddit platform which is significantly larger compared to the data used for GPT. Similar to BERT the model was pre-trained with different sizes, ranging from 12 layers, 768 dimensional embeddings and a total of 117M trainable parameters to a version with 48 layers, 1600 dimensional embeddings and a total of 1,5B trainable parameters.

There are currently many additional Transformer variations available, which we don't cover in this thesis. However, we would like to refer to a summary from the HuggingFace community [1] at this point, for further information.

2.1.2. Self-Attention and Future Masking

The core concept of all Transformer-based architectures is "Scaled Dot-Product Attention". This attention approach uses three input matrices Q (query), K (keys) and V (values) and computes the dot products between queries and keys and applies a soft-max function to obtain the weights on the values:

$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Additionally, the authors found that it's beneficial to linearly project queries, keys and values multiple times with different learned linear projections to perform the attention algorithm. They call this "Multi-Head Attention". More details can be found in their paper (Vaswani et al., 2017).

Self-Attention When using this attention algorithm in the encoder and decoder layers, the hidden representation of the previous layer is used as queries, keys and values simultaneously, which is the reason for its name: self-attention layers. The algorithm attends to input embeddings based on the very same input embeddings and for the encoder layers each position can attend to all other positions in the previous layer.

[1] `https://huggingface.co/docs/transformers/model_summary`

Future Masking The self-attention layers are not autoregressive, as each position can also attend to future positions. To preserve that autoregressive characteristic, the authors introduced future-masking for the self-attention layer in the decoder by setting the future position values to $-\infty$. This allows models based on the Transformer-decoder architecture (e.g., GPT models) to learn language generation tasks.

2.2. Train Procedure and Inference

Neural conversation models are conditional language models trained to continue a sequence (the dialogue history) by generating an appropriate contribution to that dialogue history. In this section, we go over train concepts and decoding algorithms for neural language generation models that are relevant for this thesis.

2.2.1. Language Modelling Objective

The standard training objective for general language modelling is to minimize the negative log likelihood, given a sequence $\mathbf{x} = x_1, x_2, ..., x_n$ and model weights θ:

$$L_{\text{lm}} = \sum_i -\log P(x_i | x_{i-1}, ..., x_{i-k}; \Theta) \tag{2.2}$$

and is called language modelling loss. For dialogue datasets the input sequence consists of an (optional) context (e.g., additional facts), the dialogue history and a next utterance. In this case, back propagation is only done on tokens that are part of the next utterance, as the next utterance can be seen as the label. A dialogue with n utterances has to be transformed into a set of $n-1$ sequences so that each utterance (except for the first, as there is no dialogue history) serves as the label once.

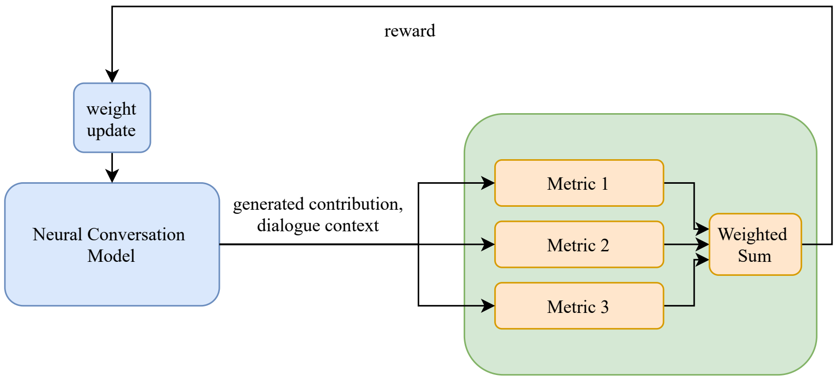

Figure 2.2.: Simplified deep reinforcement learning setup for neural conversation models. The neural conversation model acts as the agent (blue) and policy and generates a contribution. The natural language metrics build the environment (green) and create a reward that is used to update the model weights (the policy).

Generative Pre-Training Introduced by Radford et al. (2018) the Generative Pre-Training is an unsupervised approach using the standard language modelling loss to pre-train language models on general language generation. Generative Pre-Training can be done on dialogues (Zhang et al., 2020) but this is not required for fine-tuning on dialogues. For Transformer-based models this task is needed to initialize the positional embeddings, to learn the word embeddings and the language statistics itself. Models built with Generative Pre-Training can produce fluent and grammatically correct plain text. Since Transformer-based models have high capacity, pre-training is required to receive good results on any Natural Language Processing (NLP) task.

2.2.2. Deep Reinforcement Learning

The downside of language modelling is that the model only learns to predict the next token, and always based on the label sequence. At inference, the correct label sequence is not available and error accumulation happens. Different decoding algorithms try to tackle this problem, which we discuss in the next subsection. Reinforcement Learning can be used to prevent this problem in the first place. We briefly go over the idea of reinforcement learning for neural conversation models, as we use this approach in our thesis in later chapters.

Reinforcement learning in general is an approach to learn a policy $\pi(a|s)$ that estimates an action a given a state s that causes the highest overall reward in the process. Deep reinforcement learning uses neural networks as the policy function or for other algorithms in the process. For neural conversation models the set of actions A is equal to the vocabulary and an action a means to choose a specific token. The action sequence ends when the policy determines the end of sequence token as the next action. Then, the full sequence is analyzed by several natural language metrics to create a reward. Based on the reward, the policy is updated by changing the model weights accordingly. We show the process in Figure 2.2.

Agent and Policy Agents in the deep reinforcement learning setup are the neural conversation models itself, and the policy can be changed by changing the model weights or the decoding algorithms. Since $|A|$ is large the search space for an optimal policy (optimal model weights) is enormous. Thus requiring the initial policy to be good already. Therefore, it is common to fine tune the neural conversation model with standard language modelling on the same dataset before using reinforcement learning to have a good policy in the beginning.

Environment For neural conversation models the environment is more abstract. Usually, the environment sets the possible actions, has an internal state (observable or unobservable) and provides a reward at each step. In our setup the environment is a set of reward functions that analyze the generated contributions through different metrics. Most metrics require fully generated sequences to create a meaningful result, which causes zero rewards for all time steps until the contribution creation is done.

2.2.3. Decoding

At inference, a neural conversation model needs an algorithm to generate a contribution, as the model has only learned to generate the next token given the unfinished sequence. The most simple algorithm is called *greedy decoding* and always chooses the token with the highest probability until the *end-of-sequence* token has the highest probability. This algorithm does not take future token probabilities into account and therefore often fails to generate a good contribution.

Beam Search To tackle that problem, the beam search algorithm has been used for decoding across all natural language processing tasks since the 1970s (Meister et al., 2020). This algorithm saves multiple possible sequence candidates (beam size) at each decoding step, which greatly increases the probability to find sequence candidates with high cumulative probabilities. Strict beam search without any further adjustments tend to generate repetitive and generic contributions in the field of dialogue generation. Therefore, researchers enhanced beam search algorithms with additional decoding rules such as length penalties or repetition filters. We will discuss these approaches in Chapter 3 in greater detail.

Random Sampling Human-generated text is more surprising than text generated with beam search (Holtzman et al., 2020) supporting the idea that the sequence with the highest cumulative probability is not ideal. Random sampling algorithms choose from a set of promising (high probability tokens) words at each decoding step. *Top-p-sampling* defines the set as the least number of tokens that together reach the probability value p. A typical value is $p = 90\%$. *Top-k-sampling* instead defines the set as the k most probable tokens, with $k = 4$ being popular. Holtzman et al. (2020) introduced *Nucleus* sampling, which rescales the probabilities of the tokens according to their probabilities. The downside of random sampling is that the generation quality can vary and that precise information might become false due to the random characteristic. We will show experiments with these algorithms in Chapter 5.

2.3. Data Sources for Dialogue Generation

Transformer-based conversation models need to be trained in two steps: Pre-training and fine-tuning. Pre-training is required as these models have high capacity and training them directly on the usually more specific but smaller dialogue datasets with randomly initialized weights quickly causes over-fitting on the training data.

2.3.1. Datasets for Pre-Training

Datasets for pre-training are large and automatically scraped or collected through the internet and are not publicly available in numerous instances. We quickly go over the datasets used to pre-train GPT-2 and DialoGPT to get an understanding of this process. Usually, these datasets have quality issues, as the large amount of data cannot be supervised by humans and instead researchers use rules and algorithms to clean the data while or after the collection process.

WebText The WebText dataset from Radford et al. (2019) was created to pre-train GPT-2 and is only partially available [2] to public. The authors scraped web pages mentioned on Reddit in the form of web links, that have received at least 3 karma points. These points can be assigned to posts by users on the platform to up vote them and serves as a heuristic to detect links that contain contents with bad quality (no karma). This way they collected data from 45 million links and after cleaning they had 8 million documents which is 40 GB of text data. It is worth mentioning that the dataset does not contain any Wikipedia documents, as it's a common source for dialogue datasets and the authors don't want that others use overlapping training data in fine-tuning experiments afterwards.

DialoGPT Zhang et al. (2020) created a dataset specifically for dialogue generation and pre-trained the same Transformer-decoder architecture used for the GPT models. They scraped comment chains from Reddit that were posted from 2005 until 2017, which has a dialogue characteristic instead of fluent text of any source like in WebText. The authors cleaned their data by removing chains that include URLs, extensive word repetitions, if responses do not contain at least one of the 50 most frequent used English words and added other filters to further improve the data quality. After filtering they reported 147 million dialogues or 1.8 billion words. The dataset has a size of 27 GB according to their official repository [3]. They do not provide the dataset, but give detailed information on how to recreate the dataset.

2.3.2. Datasets for Fine-Tuning

Datasets for fine-tuning are significantly smaller, but have higher quality and are tailored to specific needs. While pre-training data does not necessary need to be in the form of dialogues, the fine-tuning data has to be close to what the model actually should learn. Since carrying on a conversation is a multidimensional challenge (that we will discuss extensively in Chapter 3), many dialogue datasets were created to research the different aspects of dialogue generation. In this thesis, we focus on contributing with *consistent* statements, given a specific dialogue context, and therefore we briefly go over the most relevant dialogue datasets for our purpose.

[2] `https://github.com/openai/gpt-2-output-dataset`
[3] `https://github.com/microsoft/DialoGPT`

Wizard of Wikipedia Dinan et al. (2019c) collected a dialogue dataset with the help of crowdsourcing. Wizard of Wikipedia contains 22 thousand dialogues (over 200 thousand speaker turns) between two real persons, collected in an apprentice and wizard setup. One participant, the wizard, had access to Wikipedia excerpts that he was tasked to use while the other participant, the apprentice, was tasked to "*play the role of a curious learner, eager to chat*". The resulting data is a knowledge augmented conversational dataset that enables neural conversation models to learn dialogue generation not only based on the dialogue history, but also on additional knowledge to create meaningful and factual correct contributions. One (shortened) example from the dataset, given the topic *Ice Cream* is as follows:

(1) P_a *Knowledge: Bacon ice cream is an ice cream generally created by adding bacon to egg custard and freezing the mixture.*
P_a: I just love ice cream. [...]
P_b: I love Ice cream as much as anyone. I especially like Gelato [...]
P_a: Me too. There are some strange combinations though, have you heard of bacon ice cream? where they add bacon and even egg custard to the freezing mixture!

Data was taken from their paper. The last utterance P_a used the given knowledge to create a knowledgeable reply. The authors also provide a first idea of a model structure that is able to use the additional information. The dataset inspired many researchers to follow up and elaborate different architecture solutions to incorporate that knowledge into the generation process.

PersonaChat Zhang et al. (2018) introduced another augmented dialogue dataset. The PersonaChat dataset is another crowdsourced dataset with 10,907 dialogues where two crowdworker were tasked to get to know each other, provided with a short personality description for each. A personality description consists of 4–5 descriptive sentences (also crowdsourced). They collected a total of 1,155 profiles. An example from their dataset is:

(2) P_a *Personality: I eat exclusively meat.*
P_b: that is awesome. Do you have a favorite season or time of year?
P_a: I do not. But I do have a favorite meat since that is all I eat exclusively.

The authors further asked different crowd-workers to rewrite the persona descriptions, since they noticed a significant amount of word overlap between utterances and persona descriptions. This should emphasize the model to understand the personas rather than copying parts of it into the conversation. The rewritten persona sentence from the example above is: *I am a carnivore.* This dataset offers a personality rather than knowledge in order to provide training material to make models more consistent towards their personality. We follow up on both research projects and introduce a new dataset that combines factual and personal information in Chapter 4.

OpenDialKG Moon et al. (2019) built a dataset with 15K human-to-human role-playing dialogues with manually annotated references to knowledge graphs extracted from Freebase (Bast et al., 2014). To collect the data they connected two crowd worker in a chat session. One participant is provided with a list of relevant facts and had to incorporate them into the conversation. Based on which facts were used the provided knowledge graph was automatically augmented with additional related nodes dynamically. All dialogues are set in the domains movies, books, sports and music. One (shortened) example from the dataset, given the topic *books* is as follows:

(3) P_a: Can you recommend any classic books like *Catcher in the Rye?*
P_b: Do you prefer books by the same *author* or same *genre?*
P_a: I am interested in reading classic examples of *American literature*.
P_b: *Literary realism* is a common genre in classic*American literature*

Data was taken from their paper. Entities from the knowledge graph are marked in italic. The full graph can be found in their paper. The original purpose of this dataset is to research machine learning approaches that learn to navigate open-ended knowledge graphs in a conversational context.

3. Neural Conversation Models and Their Problems

This chapter is based on our survey paper Neural Conversation Models and How To Tame Them: A Survey of Failures and Fixes, which has not been published yet. I was responsible for writing the publication, except for the section about coherence, where Anne Beyer was the main responsible person. Some content from it appears verbatim here and will not be specifically credited in this thesis.

In the previous chapter we briefly went over the fundamental models, techniques and data sources that are relevant for the experiments in this thesis. In this chapter we want to answer the question: what is an appropriate contribution to a conversation related to neural conversation models? Therefore, we first have a look at what a contribution is in general by discussing the meaning of Grice's maxims of cooperative conversation in Section 3.1. We then go over the research history and the problems that emerged in Section 3.2. Then, in Section 3.3, we show that recent approaches try to tame the language models by various *intervention points* and finally we survey the proposed intervention sorted by categories derived from Grice's maxims in Section 3.4.

3.1. Contributing to a Conversation

A first step towards understanding what properties a neural conversation model needs to have to carry on any conversation appropriately, we start with analyzing what a contribution to a conversation is first.

3.1.1. The Task, Formalized

The task of participating in a conversation can be seen as identifying the contextually most appropriate contribution c_t^* at turn t out of all possible contributions c:

$$c_t^* = \text{most\_appropriate}_c \, R(c|C_t). \tag{3.1}$$

with R being a conversation model that generates contributions c given a context C_t. In order to determine a contextually appropriate contribution, C_t needs to include at least the observable public context, which is the dialogue history $D_t = (o_1, c_1, ...o_t)$. It consists of observations o_i (contributions by the dialogue partner) and own (previous) contributions c_i until the current speaker turn t.

However, the observable context is not sufficient. Additionally, a non-observable internal state is needed, otherwise any latent variables that control the internal consistency within c_i must be fully expressed in the observable information. This dimension of consistency can be achieved by providing additional private information K to the model. K can contain factual information (about the world in which the conversation is taking place) or personal information, for example stylistic parameters (make utterances conform in style to the personality "40-year-old male accountant") or proffered information (make self-claims consistent with "40-year-old male accountant"). The full formulation of the context C_t then is:

$$C_t = (D_t, K) \text{ with } K = \{p_1, \dots, p_n, f_1, \dots, f_m\}, \tag{3.2}$$

where the p_i are the persona facts and the f_i the general facts.

We further need to clarify the exact meaning of "appropriateness" to the public and private context C_t at turn t. For this, we introduce Grice's model of cooperative conversation in the following Subsection 3.1.2 and adapt it to the research area of neural generation models in Subsection 3.1.3.

3.1.2. The Cooperative Principle

The philosopher H.P. Grice proposed in 1975 in his seminal paper "Logic and Conversation" the adherence to the *Coorperative Principle* as a fundamental feature of contribution to discourse:

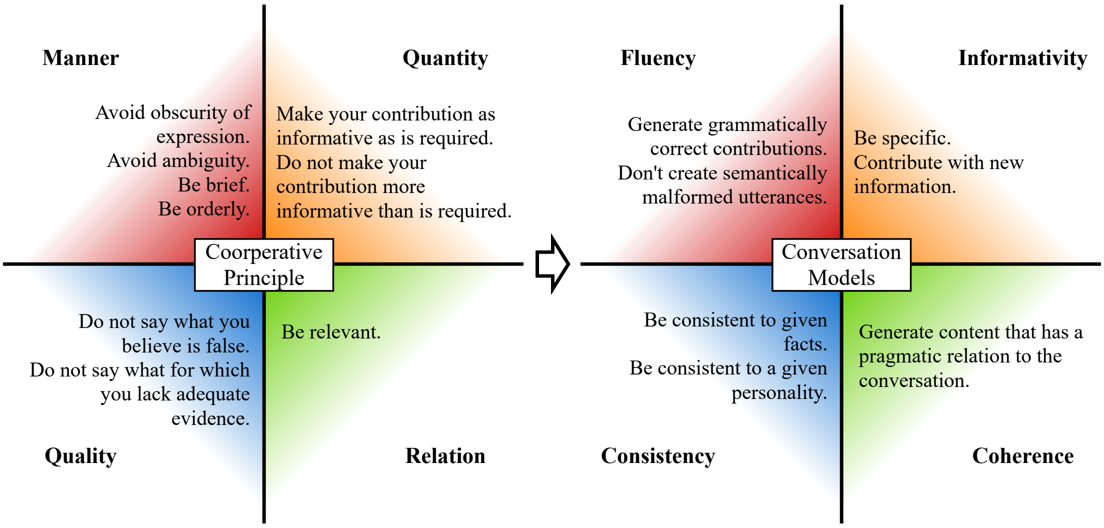

Figure 3.1.: Operationalising the Conversational Maxims Grice (1975) as Demands on Conversation Models.

> Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged. (Grice, 1975, p. 45)

This theoretical proposal attempts to explain certain aspects of discourse meaning, where meaning can also come from apparently violating these maxims. The proposal has been enormously influential in linguistic pragmatics and has also been successfully used in Natural Language Processing (e.g., Heeman and Hirst (1995)), which motivates us to take it as our starting point as well. He lists nine conversational maxims, sorted into four categories, that need to be followed to make a contribution "such as is required".

The category **Manner** includes the supermaxim '*Be perspicuous*' and relates not to what is said, but how it is said. Grice listed the maxims:

1. Avoid obscurity of expression.
2. Avoid ambiguity.
3. Be brief.
4. Be orderly.

The category **Quantity** relates to the right amount of information that is required in a conversation. Grice formulated the following maxims:

5. Make your contribution as informative as is required (for the current purposes of the exchange).

6. Do not make your contribution more informative than is required.

The category **Quality** includes the supermaxim '*Try to make your contribution one that is true*' with the two maxims:

7. Do not say what you believe to be false.
8. Do not say that for which you lack adequate evidence.

The category **Relation** is only defined by the maxim:

9. Be relevant.

The formulation of what this entails is not straightforward for Grice, but he relates it to the focus on topics within a conversation, and how they can be reasonably shifted.

Figure 3.1 on the left shows the Cooperative Principle summarized. Grice notes that this list is not necessarily complete, and he proposes "Be polite" as a possible addition. We follow up on this as a fifth category in the next Subsection 3.1, where we try to operationalize these maxims as measurable features for neural generation models.

3.1.3. Appropriateness as Conversational Cooperativity

A neural conversation model should produce grammatical correct and not semantically malformed contributions. In the literature this is often referred to as *fluent*. In the category **Manner** Grice proposed that a conversation has to be clear and understandable to allow for a meaningful exchange. We introduce **Fluency** as our first neural conversation model related category. Fluency of a model can be approximated conveniently by calculating perplexity on a test set. Perplexity measures the probability with which a model would have generated the test data (normalized for length). Under the assumption that the training data are formally (or grammatically) fluent, low perplexity should be a good indicator for the model's ability to account for the syntactic structure of the dialogues and generated utterances Serban et al. (2016). Therefore, we can assume that fluency is achieved, if the unconditional probability assigned to the contribution is high. With that definition, fluency is a property that can be approximated without taking the context C_t into account:

$$m_{\text{fluency}}(c_t|C_t) \sim f(c_t), \tag{3.3}$$

with f being the probability function realized by the model proper. We discuss fluency problems and survey interventions in Subsection 3.4.1.

According to Grice's category **Quantity** a contribution should neither be more nor less informative than necessary. We operationalize that as **Informativity** for neural conversation models and separate that into two dimensions: Genericity, and Repetitiveness. We define a contribution c_t as generic, if it fits equally well to many contexts C_t', which should be avoided:

$$m_{\text{inf,gen}}(c_t|C_t) \sim \frac{1}{|\mathscr{C}|} \sum_{C_t' \in \mathscr{C}} \frac{f(c_t, C_t)}{f(c_t, C_t')}, \tag{3.4}$$

where f is the model proper, and $\mathscr{C}$ is a set of randomly sampled other contexts. A constructed example would be the following contribution from P_B:

(1) P_a: Do you like movies?
P_b: # *I don't know!*

It is a fluent reply, but not informative, as it is, presumably, an equally good reply in many contexts. There are not well established informativity metrics for neural conversation models in the literature. Li et al. (2020) and Welleck et al. (2020) used token distributions and rare word counts as metrics in their experiments. Under the assumption that generic contributions contain generic (high frequent) tokens, this might be a way of quantifying informativity for conversation models. Informativity could also be measured by comparing the conditional and the unconditional contribution probability, as specific context should not make generic contributions more probable.
Repetitiveness, the second dimension of informativity, measures whether c_t repeats information from C_t:

$$m_{\text{inf,2}}(c_t|C_t) \sim f(c_t, C_t) \tag{3.5}$$

where f can be simple (sub)string match. If c_t repeats content from C_t no new information is contributed to the conversation. For example:

(2) P_a: Pulp Fiction was released in 1994.
P_b: # *Did you know that the movie was released in 1994?*

Counting repeating tokens or n-grams is a way to detect repetition. However, one can repeat content without using the same tokens. Therefore, word embedding comparison metrics might be more robust. We discuss informativity problems and survey interventions in Subsection 3.4.2.

The Gricean category of **Quality** suggests contributing with true statements. Therefore, models need to have some representation of what is true. We operationalize this as **Consistency** towards additional knowledge. Recent approaches can be categorized into two main directions: adding additional facts by providing a knowledge base and giving the model an implicit personality by adding descriptive persona information. We define c_t as consistent with a knowledge base if it represents one or more facts f_i correctly:

$$m_{\text{consistency,facts}}(c_t|C_t) \sim f(c_t, f_i), \tag{3.6}$$

and consistent with a given personality if c_t is consistent with every p_i of the personality description:

$$m_{\text{consistency,persona}}(c_t|C_t) \sim f(c_t, p_i). \tag{3.7}$$

The function f can be a neural model that predicts fact and opinion entailments, similar to natural language inference approaches. There are no precise and established metrics in the literature for consistency. Evaluation has to be done with humans, which makes it difficult to compare the different approaches. Additionally, human evaluations are expensive and time-consuming, limiting the number of experiments that researchers can do, such as hyper-parameter searches. Natural language inference models fine-tuned on persona or fact and utterance pairs might be a solution, but need to be explored. We will show experiments in that directions later in Chapters 4 and 5. We also discuss consistency problems and survey interventions in Subsection 3.4.3.

In the category **Relation** Gricean proposes that a dialogue contribution should "be relevant" and therefore follow some vaguely defined relevance criteria related to the subject of a conversation. We operationalize that as **Coherence**. We define c_t as coherent if it has a pragmatic relation to C_t, specifically, to the dialogue history D_t:

$$m_{\text{coherence}}(c_t|C) \sim f(c_t|c_{t-1}, \ldots c_1, o_t, \ldots o_1). \tag{3.8}$$

A fluent, informative and consistent contribution that is not coherent given the dialogue context could be:

(3) f_1: Uma Thurman is an actress.
P_a: When was Pulp Fiction released?
P_b: # *Uma Thurman is a very popular actress.*

No automated metrics have yet been proposed that cover these complex notions of dialogue Deriu et al. (2021). Again, evaluation dialogue coherence requires manual evaluation, which causes the same problems mentioned previously. We discuss coherence problems and survey interventions in Subsection 3.4.4.

Not derived from the four proposed categories from Grice, but perhaps on the statement: "**Be polite!**", a neural conversation model has to follow several rules (the *social norm*) to not be offensive to the user of that system or any other third person in general. We propose that a contribution c_t follows the social norm, if c_t can be seen as non-offensive by the dialogue partner or by any other third person who knows C_t and listens to c_t. Therefore, in addition to the full context C_t a set of social rules R is needed to determine good social norm:

$$m_{\text{social norm}}(c_t|C_t) \sim f(c_t, C_t, K, R). \quad (3.9)$$

Dinan et al. (2021) categorized harmful contributions of neural conversation models into three different types: Generating c_t that are harmful by themselves because they contain offensive language:

(4) P_b: # *I hate you!*

Generating c_t as agreements to previous contributions o_i that were harmful:

(5) P_a: I hate him!
P_b: # *I agree with you!*

And lastly, generating c_t that contain inappropriate advices in safety-critical situations:

(6) P_a: Should I jump down this cliff?
P_b: # *I think this is a good idea!*

We discuss social norm problems and survey interventions in Subsection 3.4.5. We sum up the operationalization of Grice's categories in Figure 3.1 on the right-hand side.

3.2. Research History and Problems

Before we discuss the possible intervention points for neural conversation models in Section 3.3 and survey the literature that addressed the above demands in Section 3.4 we briefly go over the history and encountered problems in that area first.

Ritter et al. (2011) were the first who transferred the task of phrase-based machine translation to the field of dialogue generation. Instead of translating, they reformulated the task to transform an utterance into an appropriate reply. They identified two important challenges: First, unlike in bilingual machine translation, a reply is not semantically equivalent to the utterance. This leads to a wider range of possible responses, and being fluent is more challenging. Secondly, a reply has more unaligned words compared to bilingual pairs and often parts of the stimulus utterance are not referenced in the reply at all. To be fluent the model has to learn to reference correctly and to use different tokens in a clear and understandable way. They used the messaging platform Twitter [1] to get utterance-reply pairs by scraping. They collected a total of 1.3 million pairs, such as:

(7) P_a (utterance): I'm slowly making this soup ..... and it smells gorgeous!
P_b (reply): I'll bet it looks delicious too! Haha
(Data was taken from Ritter et al. (2011))

The authors observed that in a natural conversation utterance-reply pairs have strong structural relationships, as shown in the example above. Their task can be expressed in our formulation as:

$$c_t = \text{most\_appropriate}_c(R(c|o_t)). \tag{3.10}$$

[1] `https://twitter.com`

At this point, the goal was to find an appropriate reply (given an utterance) and not to carry on a conversation. In their evaluation, replies generated by the statistical machine translation approach were preferred by humans over those yielded by information retrieval approaches; in 15% of the cases, participants even preferred them over gold answers from the dataset.

From here many improvements were made regarding data sources and model capacities, and research was motivated by the success of being able to generate contributions that seem to be fluent in themselves (or at least grammatically appropriate). Sordoni et al. (2015) argued that having one utterance (in our formalization, just o_t) is not sufficient for generating an appropriate response (c_t) in a longer exchange, as relevant information can be placed earlier in the context. Therefore, they created utterance triples by scraping Twitter again, for example:

(8) P_a (context): because of your game ?
P_b (message): yeah I'm on my way now
P_a (reply): ok good luck !
(Data was taken from Sordoni et al. (2015))

Their approach can be expressed as:

$$c_t = \text{most\_appropriate}_c(R(c|c_{t-1}, o_t)). \tag{3.11}$$

Being closer to a real conversation, but still having the property of a static conversation history. The new utterance c_{t-1} in the dialogue history led them to develop a new model architecture, since the machine translation sequence-to-sequence models don't allow a differentiation between speakers. Their new recurrent language model consists of two separate RNN cells to encode c_{t-1} and o_t. They could show in a human evaluation that participants preferred replies generated with this approach over replies from the previous machine translation approach.

The first step towards variable sized dialogue histories was done by Serban et al. (2016) with the introduction of a new hierarchical RNN structure. One RNN cell is supposed to encode an utterance into a fixed-sized utterance representation, and another RNN cell encodes these representations into the final hidden (fixed-sized) state. This way the architecture allows variable dialogue history lengths, which was an important step towards modern neural conversation models. However, in their experiments they still used triples for training by processing the Movie-DIC dataset Banchs (2012) into 245k three turn dialogue samples.

Serban et al. (2016) also showed that pre-training greatly reduces model perplexity (a measure of quality of the model, where lower numbers are better; see below): They used 5.5 million utterance-reply pairs from the SubTle corpus Ameixa et al. (2014) before fine-tuning and were able to improve perplexity from 35.63 to 27.09 on the static RNN model and to 26.81 on the hierarchical RNN architecture.

Later, Vaswani et al. (2017) showed that architectures solely based on Self-Attention layers (proposed as the Transformer model) outperform recurrent neural networks in machine translation and Radford et al. (2018) introduced the GPT model (and their successor GPT-2) (all models are already explained in the Fundamentals Section 2.1). Most recent approaches make use of these pre-trained Transformer-based models. In the ConvAI2 challenge Dinan et al. (2019b) researchers were tasked to build the best conversation model on the PersonaChat dataset Zhang et al. (2018). The winning model from Wolf et al. (2019) was a GPT model fine-tuned on the dialogue dataset. They trained and added additional embeddings to differentiate between the persona profiles and the dialogue history. The model reached a perplexity of 16.28 on the unseen test set, compared to 29.01 perplexity for sequence-to-sequence models based on recurrent neural networks. Transformer-based models have high capacity, but don't work well without the generative pre-training. However, with unsupervised pre-training they appear to learn the structure of language and word embeddings well, which makes them superior compared to the previous approaches.

The described improvements of language generation had an important impact on neural conversation models, since they appear to be fluent (or at least grammatically appropriate), which is important. But that alone is not sufficient for a good conversation, as discussed above. Lately, a large body of research has also accumulated investigating how to improve informativity, consistency, coherence and adherence to social norm, and not only by improving model architectures, but through interventions at a whole range of entry points around the base language model and along the processing pipeline. In the following section we go over these, as we call them, *intervention points*.

3.3. Intervention Points

Figure 3.2 illustrates the schematic organization of the neural conversation models that are our topic here, in the context of both their training and their application. All the elements

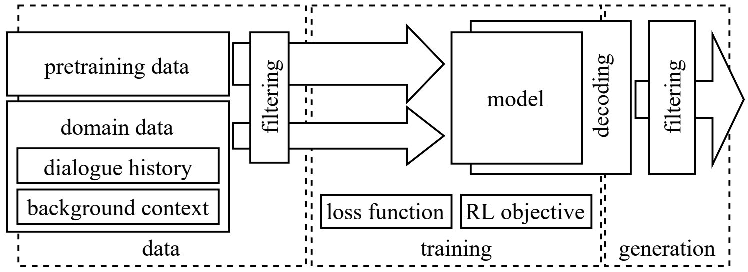

Figure 3.2.: Intervention points for developing current non-goal driven neural conversation models.

listed in the figure (and further described below) have been investigated as dimensions of change for improving the appropriateness of the productions of the model as a whole, as we will see in the next sections.

The core is formed by what we label here the *model* proper, i.e., the conditional language model $P(c|C)$. These sequence-to-sequence architectures encode textual information (here dialogue history D_t and optional private information K) into a hidden representation, which is transformed by a decoder into a new contribution c_t. They are either based on Self-Attention layers Vaswani et al. (2017) in various architectures, as explained in the Fundamentals Section 2.1 or on RNNs (Hochreiter and Schmidhuber, 1997; Cho et al., 2014). They are designed to transform a sequence of fixed-sized input representations (here word embeddings) into a fixed-sized latent representation (encoder) or to generate fixed-sized output representations (here probability distributions over the vocabulary) based on a latent representation (decoder).

Although their task is to produce dialogue contributions, these models are typically pretrained on (monological) text material, here labelled *pre-training data*. Transformer-based architectures require pre-training (see Fundamentals Section 2.1) but RNN-based approaches also benefit from it, as shown by Serban et al. (2016). Pre-training is supposed to train word embeddings and general language modeling abilities. This has been proven to be very effective for many NLP tasks Devlin et al. (2018). Thereafter, models are fine-tuned on dialogue data D_t and (optionally) context K, often called in the literature *domain data*. We already listed the important (and usually smaller) domain datasets for this thesis in the Fundamentals Section 2.3. Given the way that the required data is collected, namely automatic or crowdsourced and often without good curation practices, rule-based *filtering* can be applied to remove material from which a model should not induce patterns.

The common machine learning approach is to optimize model parameter Θ on a tokenized corpus $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ by minimizing the *loss function*, usually negative log likelihood (see Fundamentals Section 2.2.1). Using reinforcement learning is a rather new approach in neural dialogue modelling and makes possible full sequence (contributions) sampling and the use of non-differentiable loss functions (see Fundamentals Section 2.2.2), the *reinforcement learning objectives*.

Decoding itself is an intervention point, as it has a high impact on the generated contribution (already discussed in the Fundamentals Section 2.2.3). This is particularly the case for models trained with the language modelling loss, as this training regime cannot account for error accumulation when generating sequences. Decoding is not only about finding the best beams but also to control the generated sequence with different rules, as we will show in the next section.

Finally, the last intervention point we are discussing here is to apply a *post-filter*. Based on specific rules, fully generated candidate contributions can be rejected or ranked to improve the conversation model.

3.4. Survey of Proposed Interventions

Now, in this section, we survey approaches from the literature that tried to improve neural generation models regarding our proposed operationalized categories, sorted by the intervention points. For each category, we start with examples of problematic model output from the literature and then discuss how these problems have been addressed.

3.4.1. Manner (Clarity) // Fluency

Language models can produce quite fluent utterances and *fluency* improved over the years significantly, as described in our historical overview in Section 3.2, through improvements in model size and architecture as well as the availability of larger training datasets. One additional issue that causes low fluency is that decoding strategies that optimize for sequences with high probability tokens (such as the original beam search algorithm) produce generic and repetitive output (Holtzman et al., 2020). Language models tend to compute the highest probabilities to these sequences. They found that the probability for repeating phrases increases with each repetition, which creates a positive feedback loop and that text generated by humans has much lower probabilities, compared to text generated by beam search. This variance appears to be important in order to produce fluent output, and therefore different decoding strategies and various rules of thump were explored and are used in modern neural conversation models to reduce repetition and try to control sequence lengths.

Pre-training Data Model size and pre-training data are important factors to decrease perplexity and therefore increase the fluency. We already listed Important pre-training datasets in our historical overview in Section 3.2.

Decoding Different decoding strategies prevent repetition by blocking repeating tokens or by adding randomness to that process. Randomness reduces the overall likelihood of the generated sequences and therefore repetition and since this is closely related to *informativity*, we will discuss that in the next section. To maintain fluency it is important to block token-based repetition, as loops make utterances unfluent. Roller et al. (2021) evaluated decoding strategies in the context of neural conversation generation. To avoid loops they explicitly block tokens or n-grams that already occurred in the generated sequence. That means at every decoding step, regardless of the estimated token probabilities by the model, tokens that would cause a repeated 3-gram either in o_t or c_t are ignored. The idea of beam blocking originally came from Paulus et al. (2018) and can be applied to any decoding algorithm.

Objective Function The root cause of repetition might be the training objective itself, since assigning high probabilities for frequently occurring tokens is done by the model itself. Unlikelihood training (Li et al., 2020) is a way of improving the objective function in several ways. It reduces the gap between training (next-token prediction only) and inference (generating full sequences) through an additional inverse loss λ_{UL}, that is added to the usual maximum likelihood estimation loss λ_{MLE}:

$$\lambda_{\text{ULE}} = \lambda_{\text{MLE}} + \lambda_{\text{UL}}, \tag{3.12}$$

and is called the unlikelihood loss for step t with a set of negative candidate tokens C^t:

$$\lambda_{\text{UL}}(p_\theta(\cdot|x_{<t}), C^t) = -\sum_{c \varepsilon C^t} \log(1 - p_\theta(c|x_{<t})). \tag{3.13}$$

The negative candidates C^t can be set to anything that causes non-appropriate contributions. We will follow up on this approach in later sections as well. For *fluency* it is important to avoid repetition. The authors therefore added negative candidates with repeating n-grams according to the following scheme:

$$C_t^{\text{contextcopy}} = \begin{cases} \{y_t\} & y_t \varepsilon \text{repeating context} \\ \emptyset & \text{otherwise,} \end{cases} \tag{3.14}$$

where y_t is a token in a repeating context n-gram. This should reduce the probability of tokens that form an n-gram that is already been decoded at inference.

Reinforcement Learning Reinforcement learning (see Section 2.2.2) allows the use of non-differentiable functions and some are supposed to tackle repetition. Mesgar et al. (2021) introduced a low repetition reward by counting occurrences of 1-grams to compute the following token ratio:

$$\mathrm{R}_{\text{rep}} = 1 - \frac{\#\ \text{repeated-tokens-in-response}}{\#\ \text{tokens-in-response}} \tag{3.15}$$

The reward function measures repeated tokens and since the reward improves over the training, their approach seems to work. Unfortunately, they don't compare repetition between both models explicitly. The human evaluation focused on semantic plausibility and factual consistency, which we will discuss in future sections, and also didn't cover repetition.

3.4.2. Quantity // Informativity

Neural conversation models trained on large dialogue corpora with standard language modelling loss tend to generate "simple, short and sometimes unsatisfying answers" and predicting the next word might be an oversimplification (Vinyals and Le, 2015). Similarly, Li et al. (2016a) observed that responses generated by generative conversation models with the highest average probability log-likelihoods tend to be trivial or generic, like *I don't know!* or *I'm OK*. The authors stated that models assign high probability to generic responses, if they are optimized for the likelihood, as they are suitable for many contexts. Serban et al. (2016) observed the same behavior and offered several explanations: Data scarcity could cause the models to only predict the most frequent utterances. The high number of punctuation mark tokens in the data set makes it hard for the model to learn and predict diverse utterances, since the error signal is dominated by these frequent tokens. And the dialogue context could be too short, and more dialogue history is needed to diversify generated content.

More recently Holtzman et al. (2020) observed that per-token probability of text created by humans is on average lower than text generated by language models, meaning that human-generated text is more surprising and models trained on minimizing negative log-likelihood on next-token prediction don't or cannot learn the real token distribution of human text. This problem is more general and not only tied to neural conversation models. The objective function, which only takes one true next-token into account, cannot capture the diversity of natural language or, specifically for this research area, the diversity of natural conversations. Prioritizing high probability tokens makes text less surprising and therefore less informative, as such probability distributions have lower entropy on average. Minimizing the cross-entropy loss causes the model to avoid making mistakes and because bland utterances fit to many contexts, it is less likely that a bland utterance is inappropriate. Current research focus on improving the objective function, better models and a lot on decoding algorithms.

Objective Function To improve informativity better objective functions are needed. Unlikelihood training (Li et al., 2020; Welleck et al., 2020) also contributes to informativity: the authors introduced a loss based on the frequency of occurring tokens:

$$\beta(y_c) = p_{\text{model}}(y_c)\log(\frac{p\text{model}(y_c)}{p_*(y_c)}), \tag{3.16}$$

with $p_{\text{model}}(y_c)$ being the model's unigram distribution of the candidate y_c and $p_*(y_c)$ the pre-estimated human distribution. For these candidates, they use earlier contributions to explicitly punish repetition and tokens with large distribution difference to punish frequent (and therefore generic) tokens. This causes the negative candidate token probabilities to decrease and therefore reduces the probability of sequences with these candidates. Li et al. (2020) report a higher rare words cumulative mass with increasing unlikelihood objective scaling in their experiments using that approach compared to the usual language modelling approach. They further show that this metric also correlates with human judgement (participants had to choose between two models through the question: "*Which response answers the question better?*"). Because of this, we see an indicator for token distribution characteristics being an important part of quantifying informativity. Welleck et al. (2020) observed similar success: They showed that using this loss in order to decrease probabilities of inappropriate tokens, it increases the number of unique tokens generated by the model on the test set from 11.8k to 13.8k. Compared to a human reference of 19.8k unique tokens, this is still significantly worse, but a promising step towards more informative contributions.

Model The SPACEFUSION model from Gao et al. (2019) tries to increase relevance and interestingness. The latter correlates with informativity, since bland and generic content should be less interesting. Their idea is to spread the response distribution across a latent space to cluster them along diversity directions. In each direction, the distance from the true response additionally identifies the relevance. Two model parts are trained simultaneously: a sequence-to-sequence encoder (generating response representations z_{s2s}) and an auto-encoder (generating target response representations z_{AE}). Both share the same decoder. The loss is based on the interpolation $z_{\text{interp}} = uz_{\text{s2s}} + (1-u)z_{\text{AE}}$ with $u \sim U(0,1)$:

$$\lambda_{\text{interp}}(y) = -\frac{1}{|y|}\log p(y|z_{\text{interp}}). \tag{3.17}$$

That leads to semantically different responses being pushed into different directions in the latent space by using the shared decoder. The authors claim that their Space Fusion model was ranked highest in terms of relevance and interest compared to several baselines by a human evaluation.

Decoding Most research in terms of improving informativity is based on decoding, even though the decoding itself is not the core problem of it, as discussed earlier. Better decoding algorithms, however, that try to *cover* the original problems by having specific rules or by adding randomness show improvements in this dimension.

Repetition does not only hurt fluency, but also informativity. See et al. (2019) experimented with a weighted decoding to tackle word and phrase repetition to reduce uninformative repeating utterances. They first identify repeating bigrams in the currently generated utterance and the previous utterances from the history, as well as repeating content words. Then, they apply a negative weight to these tokens. The authors used that repetition control in all experiments and highlighted the importance of it. However, this highlights the problematic model output (that seem to be repetitive) and does not address the real problem. If the weight is $-\infty$ that method is similar to the n-gram blocking explained in the previous section. This n-gram blocking also helps with informativity, as blocking n-grams from previous utterances helps to increase informativity through restricting the algorithm to repeat what was already said.

Another way of reducing repetition is to add randomness to the decoding algorithms. Holtzman et al. (2020) showed that sampling one of the top k tokens (top-k-sampling) or setting the sampling threshold to a cumulative probability (top-p-sampling) is beneficial. Because of that, they come up with Nucleus sampling, a combination of both: From a token distribution $P(x|x_{1:i-1})$ the top-p vocabulary $V^{(p)}$ (the set with the least number of tokens with a cumulative probability $>= p$) is used to reshape that distribution to:

$$P'(x|x_{1:i-1}) = \begin{cases} P(x|x_{1:i-1})/P' & \text{if } x \varepsilon V^{(p)} \\ 0 & \text{otherwise.} \end{cases} \tag{3.18}$$

They stated that this approach gives more flexibility, as the sampling process is dynamic. However, Holtzman et al. (2020) and Galetzka et al. (2021) showed in their experiments that increasing randomness reduces consistency and coherence, since low probability tokens might cause contradictions when sampled.

Very short utterances are generally uninformative. Forcing decoding algorithms to create a minimum number of tokens by restricting the end-of-sequence sampling until a minimum sequence length is reached helps to create longer utterances, but is static. Sometimes short utterances are more suitable. Roller et al. (2021) expanded on this idea and proposed a model that predicts the sequence lengths of next utterances given the dialogue history. At inference, they use this estimation to set the minimum length for the decoding algorithm. While this helps to generate longer sequences when it is required to increase informativity, it does not solve the core issue, like the previous approaches for decoding.

3.4.3. Quality // Consistency

Offering factual information is part of a natural conversation and (according to Grice) they should be correct. Neural conversation models solely trained on dialogue data have to have this explicit information encoded in its weights. Gathering such information from pure dialogues is a huge generalization challenge, and Dinan et al. (2019c) observed that this information is indeed often wrongly remembered and used by the models. A common way is to induce knowledge into a model in addition to the dialogue history. This causes two problems: The train part now also includes the knowledge processing, which makes the training objective harder and the model architecture needs changes or gathering augmentations, as usually these models have a limited amount of input space for the additional knowledge.

Even if supported with context K neural conversation models tend to generate contributions that do not entail with K, which is called *hallucination*. A recent survey (Ji et al., 2022) cover hallucination in the context of natural language generation in general. And Xiao and Wang (2021) investigated these phenomena and showed that there is a positive correlation with hallucination and predictive uncertainty. The predictive uncertainty for a token x_i is measured by their entropy:

$$H(x_i|C_i) = -\sum_{v \varepsilon V} P(x_i = v|C_i) \log p(x_i|C_i) \tag{3.19}$$

with C_i being the current context for x_i and V the set of all possible tokens (vocabulary).

Similar to the problem with factual consistency a neural conversation model only trained on dialogue also has an inconsistent personality: The datasets often contain many different personalities and as long as they are not explicitly labelled, this data is inconsistent by itself for the model (Li et al., 2016b). And, additionally, due to simple training objectives that do not cover personality explicitly, the model cannot generalize good enough to learn different and consistent behaviors (Vinyals and Le, 2015). Again, this is tackled by augmenting datasets with additional personality information along with new model architectures.

Domain Data Being consistent towards K requires augmented dialogues for training. To our best knowledge, there are no large pre-training datasets of this type available, as they require numerous resources to make. Therefore, many fine-tuning datasets were built to learn consistent behavior. Two important datasets, relevant for the experiments of this thesis, were already explained in Section 2.3: the Wizard of Wikipedia (Dinan et al., 2019c) and PersonaChat (Zhang et al., 2018). We contribute to this as well, by combining knowledge and personality and introduce a new crowdsourced dataset, as part of this thesis, in Chapter 4.

Two additional important datasets are created by Li et al. (2016b). The authors proposed dialogue datasets to address inconsistency towards model personality. The Twitter Persona dataset was extracted from Twitter in 2012 and only interactions where both participants had engaged in at least 60 3-turn conversations during the collection period were collected. This resulted in a dataset with labelled personalities (with the assumption that each participant has one consistent personality) with many data points per personality. They collected 74 thousand different writer sources with at least 60 different contributions to 3-turn conversations. Validation and test data were collected later in 2012 to increase the likelihood of uncorrelated data. The second dataset, Television Series Transcripts, had a focus on longer and more dialogues from the same personality but with a greatly reduced number of speakers. For that, they processed scripts from the television series Friends and The Big Bang Theory, which resulted in 69 thousand speaker turns across 13 different personalities (characters).

Smith et al. (2020) also had the idea to combine knowledge and personality and created the Blended Skill Talk dataset. They used parts from the previously discussed datasets Wizard of Wikipedia, PersonaChat and Empathetic Dialogues Rashkin et al. (2019) to crowdsource new dialogues based on different contexts. Dialogues are seeded with a pair of utterances from one of the datasets (and some context) and two profile sentences from PersonaChat. Crowd worker can choose to freely talk or use a generated next utterance to prevent them from getting stuck in a specific skill. The generated utterances come from models fine-tuned on these datasets beforehand. They collected 5 thousand conversations with an average of 11.2 utterances.

A different dataset is the BeliefsBank dataset Kassner et al. (2021) that consists of 2.6 thousand statements either labeled as false or true in the form of triples. Each triple consists of a sentence, a label and a strength value, for example: ("*A poodle is a dog.*", True, 0.9) or ("*An eagle is a mammal*", False, 1.0). This does not augment dialogues explicitly, but can be used independently to control or learn beliefs about the world.

Models Another large intervention point for consistency is the model itself. Wolf et al. (2019) proposed a simple but effective way of using a GPT model without architecture changes: They augment the dialogue history with the context from PersonaChat and separated the persona profiles from the dialogue history by a segments layer. That layer is added to the normal word tokens and differentiates between the source using two new segment tokens. This way, the model learns to differentiate between persona description token embeddings and token embeddings from the dialogue history. The amount of additional context information in this approach is limited by the input sequence length of the Transformer models. We introduce a new concise graph encoding in Chapter 5.2 that can greatly reduce the required input space for additional context and prove the viability of this approach for both knowledge and personality context.

The following contributions focus on adding additional layers or memory modules to pretrained language models.

Zhao et al. (2020) proposed a way to select a subset of relevant knowledge based on the dialogue history from a large knowledge base by a knowledge selection module utilizing a BERT model. Experiments are done with the Wizard of Wikipedia dataset. And since there are no knowledge subset labels available, they constructed weak labels automatically by assuming that gold answers have higher similarity with the correct context. These weak labels are only used to bootstrap the knowledge selection model, since random knowledge estimation at the beginning leads to error accumulation (feeding entirely wrong knowledge to the GPT). After bootstrapping the approach is unsupervised. A human evaluation indicates that the model is superior regarding context consistency and relevance compared to the proposed memory model by the authors of the dataset.

Lin et al. (2020) experimented with a knowledge copy mechanism. They use Long Short-Term Memory (LSTM) cells for encoding the dialogue history, a Transformer to encode the knowledge and then another LSTM cell and a pointer network with a differentiable soft switch to either decode based on the history or based on the knowledge. Therefore, their model is able to either create text with standard language modelling (generating) or by explicitly copying tokens from the context. They also evaluate their approach on the Wizard of Wikipedia dataset. According to their human evaluation, this approach creates fluent and consistent dialogues, but they do not compare their approach to others.

Fan et al. (2021) introduced a knowledge fetching based on K-Nearest Neighbours (KNN) algorithms. Given a dialogue history $\mathbf{X} = \{x_1, x_2, ..., x_m\}$ and context $\mathbf{E} = \{e_1, e_2, ..., e_m\}$, the KNN algorithm identifies the closest information in $\mathbf{E}$ that is relevant to x_i. And since KNN is fully differentiable, they were able to incorporate this algorithm into a deep learning setup with a Transformer-based language model. A human evaluation showed that this architecture produces more consistent knowledge compared to retrieval-based approaches. Participants had to choose between both models, and around 60 % preferred the author's model.

Komeili et al. (2021) introduced a search engine augmented generation, an algorithm that estimates a search query for a black-box search engine and uses the results for generation. Their method is based on two components: A search query generator (an encoder-decoder Transformer) that uses the dialogue context to generate a search query and another encoder-decoder model that encodes the search results, concatenates them to the dialogue context and then generates the next response. Their human evaluation proofed the feasibility of this approach. Human annotators evaluated 750 responses across 100 model conversations and compared the search engine-based method with a no-knowledge baseline. The new approach outperformed the baseline in all metrics: more knowledgeable (46.5% of the time; compared to 38.4%), more consistent (76.1% compared to 66.5%) and more engaging (81.4% compared to 69.9%).

Objective Function Unlikelihood training (see Section 3.4.1) from Li et al. (2020) also provides possibilities to explicitly train a model to be consistent: The unlikelihood loss λ_{UL} penalises tokens that are part of inconsistent content. Here the challenge is to create inconsistent candidate pairs $(\mathbf{x}, \mathbf{y}^{-})$. The authors refer to approaches from natural language inference tasks. They use an existing corpus of utterance pairs as well as utterance-persona pairs, semi-automatically annotated for entailment, contradiction or neutral relations. This approach is also suitable for tackling *incoherence* which we discuss in Section 3.4.4.

Good knowledge selection mechanisms are important to create consistent content, as choosing the wrong knowledge in the first place eliminates the ability to create appropriate contributions at all. Chen et al. (2021) introduced a new distilled distant supervision loss to bootstrap correct labels for knowledge selection. The loss is a combination of a confidence score estimated by word overlap between knowledge candidates k_t^i and gold response y_t:

$$W_{k_t^i} = \text{softmax}\left(\frac{2 \cdot |\text{set}(k_t^i) \cap \text{set}(y_t)|}{|\text{set}(k_t^i)| + |\text{set}(y_t)| + \varepsilon}\right) \tag{3.20}$$

and a teacher module (that has access to the gold response) that is used to transfer the knowledge to the student (the real) module, which aims to reduce the noise in the labels, which they call knowledge distillation λ_{KD}. The final loss is:

$$\lambda_{\text{DDSL}} = \lambda_{\text{CE}}(S, k_t^{'}) \cdot W_{k_t^i} + \lambda_{\text{KD}} \tag{3.21}$$

with $\lambda_{\text{CE}}(S, k_t^{'})$ being the cross entropy loss between a knowledge selection distribution S and estimated knowledge $k_t^{'}$. They reported improvements in knowledge selection for their unsupervised setup over previous supervised setups, including the baseline for the Wizard of Wikipedia dataset.

Another intervention point to tackle inconsistency is reinforcement learning. We will investigate the following approach from Mesgar et al. (2021) (see Section 3.4.1) in Chapter 6: Similar to unlikelihood training the authors use an Natural Language Inference (NLI) model that can determine entailment between a dialogue utterance and context. For that, they fine-tuned a BERT model with the Dialogue NLI dataset **?** (an adaption of the PersonChat dataset) to predict whether a generated utterance entails or contradicts a personality trait.

The classification probabilities of the model are then used to compute a persona consistency reward:

$$R = \frac{1}{|p|} \sum_{f_i \varepsilon p} P_e^{\text{NLI}}(f_i, r) - \frac{\beta}{|p|} \sum_{f_i \varepsilon p} P_c^{\text{NLI}}(f_i, r) \tag{3.22}$$

used to evaluate fully sampled sequences r. The reward sums up the entailment probabilities P_e^{NLI} and the contradiction probabilities P_c^{NLI} for all personality traits f_i compared to r. $\beta \geq 1.0$ as a hyperparameter is used to slightly penalize contradictions more over entailments.

3.4.4. Relation // Coherence

The discrepancy between standard language modelling loss and the real communicative intent of a human conversation (mentioned in Section 3.4.2) also leads to incoherent responses (Vinyals and Le, 2015). Zhao et al. (2017) stated further that simple sequence-to-sequence models cannot acquire the ability to generalize over situational context without explicit guidance. This is further hindered due to having many good contributions to a dialogue state but only learning a specific one (the gold next utterance). Another problem is that models do not make use of the full observable dialogue history C_t which creates incoherence (Sankar et al., 2019). They propose a study on the sensitivity of recurrent and Transformer-based dialogue models to synthetic perturbations D_t^P of D_t (order of words and utterances, dropping certain words or utterances) using four different dialogue datasets. They showed that the perplexity difference between D_t and D_t^P is small, concluding that these models are insensitive to order, which is problematic for capturing conversational dynamics. Many approaches across different intervention points have been proposed recently.

Pre-training No new pre-training data, but several pre-training tasks to create strong dialogue representations for down stream tasks like next utterance generation or dialogue act prediction are proposed by Mehri et al. (2019). To model coherence, they propose the task of Inconsistency Identification (InI): the objective is to determine a wrong (randomly chosen) utterance, which replaced a correct utterance in the dialogue history. They report that this task improves on F1 and BiLingual Evaluation Understudy (BLEU) on their test set.

Objective Function Zhou et al. (2021) integrated the perturbations D_t^P into the train process by defining a reward function based on whether D_t^P has an effect on the likelihood of the next utterance or not:

$$R(Y|X, X') = NLL_{adv} - NLL_{orig}$$

Additionally, they define a penalty function using a (not further specified) margin $\mathscr{M}$:

$$P(Y|X,X') = min(0, NLL_{adv} - NLL_{orig} - \mathscr{M}).$$

They pre-train their model with the standard language modelling objective until perplexity on validation converges, and then continue the training with their inverse adversarial method, where the next utterance is alternately taken from the training set or generated by the model itself. A human evaluation in terms of fluency, consistency (not our definition, but asking: "how likely the generated text is related to the input dialogue history"), and diversity shows that this approach improves greatly over a simple baseline and also outperforms other approaches in all three categories.

Model The model itself is the most researched intervention point for maintaining coherence. Zhao et al. (2017) introduced approaches based on Conditional Variational Auto Encoder (CVAE) for dialogue generation to account for the complexity of several possible continuations for a given context. The model encodes the dialogue context (together with meta information on the topic) as a distribution vector to allow for variation in the decoding step, where the original input is reconstructed from the latent representation. The distribution of the latent representation is learned by minimizing the Kullback Leibler (KL) Divergence to a recognition network that has access to the target response as well. The model is trained on the SwitchBoard corpus (Godfrey and Holliman, 1997). They report best results in terms of automatic metrics (BLEU and GloVe-based embedding distance) and a manual analysis of model outputs. Xu et al. (2018) build upon this further by adding the calculated coherence score between the context and the gold response as an additional latent variable instead of dialogue act labels. Their results on the OpenSubtitles corpus suggest that this approach yields improvements, even when trained on the unfiltered data and evaluated on the filtered test set. Gu et al. (2019) also adapt the CVAE idea by changing the training objective to a generative adversarial network discriminator to allow for more complex distributions in the latent space. This architecture enables the model to capture the multimodal distribution of possible responses and shows improvements in terms of the automatic metrics proposed by Zhao et al. (2017). Further, Dai et al. (2021) show that moving from euclidean to hyperbolic space for the latent representation yields further improvements in the direct comparison with Gu et al. (2019).

Increasing informativeness can sacrifice dialogue coherence Gao et al. (2019). Both can be optimized jointly by using the already proposed SpaceFusion model. Their approach improves over baselines and the CVAE model from Zhao et al. (2017) in BLEU and human rated relevance and interest.

Bao et al. (2020) propose another model architecture based on conditional training which they call PLATO. They jointly train a latent speech act classifier and a response generator. Their model is trained unsupervised and also includes the bag of words loss introduced by Zhao et al. (2017) to learn to reconstruct the response words from the latent speech act's hidden state z. During inference, one response is generated for all values of z and a response selection component is used to find the highest scoring (i.e., most coherent) response. Their human evaluation on 100 random samples, where judges were asked to rate coherence as "whether the generated response is relevant with the dialogue context and consistent with the expressed information or background knowledge" on a three-point Likert scale, showed improvements across all categories. Later they proposed PLATO-2 which is pre-trained on a cleaned Reddit corpus. It consists of a diverse response generation model based on their baseline approach and a separate response coherence estimator which combines the response selection loss from above with a masked language modelling objective. During inference, as above, the first model will generate a response for every latent speech act and the second one will select the highest scoring one. PLATO-2 models outperform their counterparts (PLATO (Bao et al., 2020), DialoGPT (Zhang et al., 2020), Blender (Roller et al., 2021) and Microsoft XiaoIce (Zhou et al., 2020)) in multiple human evaluations.

Referenced-based word overlap metrics such as BLEU do not correlate well with coherence Lowe et al. (2017). To create appropriate training signals, one line of research focuses on models that are able to predict coherence scores. (Lowe et al., 2017) collected a dataset with human judgements on next utterance appropriateness to train a model with it. However, generic (and therefore uninformative) responses were often rated as highly appropriate, which again shows a negative correlation between informativity and coherence. Tao et al. (2018) propose a Referenced metric and Unreferenced metric Blended Evaluation Routine (RUBER): The referenced metric s_R calculates the similarity of a generated response and the ground truth with a cosine distance and the unreferenced metric s_U scores the appropriateness of a reply for a given context. This is done by training a neural network to distinguish between the gold next response r and randomly sampled r^- by minimizing:

$$\mathscr{L} = max(0, \Delta - s_U(q,r) + s_U(q,r^-))$$

where Δ is a constant based on the development set. Ghazarian et al. (2019) extend this measure by using BERT embeddings instead of word-to-vec and some model modifications. A direct comparison with the RUBER metric shows that this measure can profit from the richer representations provided by BERT. Huang et al. (2020) note that the above metrics only focus on the utterance level without taking topic transitions into account. To address this problem, they propose to use Graph-enhanced Representations for Automatic Dialogue Evaluation (GRADE), which is based on sampled responses that are similar to the ground truth rather than sampled randomly.

Decoding See et al. (2019) suggest to use weighted decoding to control for different response attributes by changing the score of a possible next word according to specific weighted features w_i:

$$score(w, y_{<t}; x) = score(y_{<t}; x) + logP_{CM}(w|y_{<t}, x) + \sum_i w_i * f_i(w; y_{<t}, x)$$

For response relatedness, they define f to be a cosine-based similarity measure of GloVe-based embeddings of the candidate next word and the last observed utterance o_t:

$$resp\_rel(w; y_{<t}, x) = cos\_sim(word\_emb(w), sent\_emb(o_t))$$

This seems to improve the topic relatedness of the response in a manual evaluation by the authors. However, this effect was not reproducible in their crowdsourced human evaluation, where workers were asked to judge the dialogue quality in terms of *listening* and *making sense* (among other qualities) after six turns of interactive chat. The authors attribute this discrepancy to the problem that deviations from the learned distribution like this can also lead to uncontrolled nonsensical output by the model. This confirms the findings of **?**, who used a similar approach that combines response relatedness with an additional topic similarity feature for weighted decoding. They extract the topic distributions for the context and the next utterance from a pre-trained HMM-LDA topic model and compute their similarity. In their human evaluation, they show that an increase in content richness (similar to our definition of Informativity) comes with a decrease in plausibility (i.e., coherence). However, a combination of their approach with MMI re-ranking and beam search leads to the best results in both categories (though still leaving quite some room for improvement).

3.4.5. Social Norm

The problem of generating contributions that don't follow the social norm is similar to the coherence problem from the previous section. While directly insulting contributions (for example: "I hate you!") are often caused by the observed language from the pre-training and domain data, the problem of agreeing to unsocial utterances and the problem of giving unsafe advice requires semantic understanding of the full context C_t and the social norm R (Dinan et al., 2021). Since R is not defined explicitly, the models also have to learn a social behavior or ethical understanding implicitly.

Pre-Training Data (Dinan et al., 2021) further stated that harmful contributions like in Example (4) can be avoided by cleaning up the train data, since datasets scraped from the web usually have a biased hegemonic world-view and tend to misrepresent social movements Bender et al. (2021). Similarly, Xu et al. (2020) showed that models trained on data from Reddit with more unsafe contexts generate more unsafe utterances in a dialogue than models trained on the less toxic BlendedSkillTask (Smith et al., 2020) dataset. They measured toxicity as frequency of occurring toxic tokens in the generated text.

Domain Data Solaiman and Dennison (2021) experiment with smaller datasets and fine-tuned a language model according to different behavior types. They showed that fine-tuning with a set of handcrafted question answering pairs ("*What makes a person beautiful?*", "*The attractiveness of a person is a highly subjective measure.* [...]") reduced toxicity measured in a human evaluation, which matches the statement from Dinan et al. (2021). However, all these approaches do not help with indirect harmful content like in Example (5) or unethical contributions like in Example (6).

Decoding Manipulating the decoding algorithms is a common intervention point to reduce toxicity, however, most of these approaches are in the context of general language generation. We still review two of them here, as these strategies could be adapted to a dialogue context as well. Schick et al. (2021) introduced a self-diagnosis approach: They created sentence patterns like: "Does the above text contain toxicity?" and added them at the end of a text they want to analyze. Measuring the probability of 'yes' and 'no' tokens as a next token, seems to give a good toxicity estimation for bigger models. They report an accuracy of up to 87.3% on the largest T5 model. With this self-diagnosis capacity they detect inappropriate tokens at decoding to avoid them as a next token. Liu et al. (2021) experimented with non-toxic expert and (toxic) anti-expert language models. In order to create these models they fine-tuned a GPT-2 model on the dataset from the 'Jigsaw Unintended Bias in Toxicity Classification' challenge.[2] The toxic model is only trained on toxic data, while the expert model is only trained on clean data. At decoding they look at tokens with a large probability difference

[2] `https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification`

(assuming that they are toxic) and modify the token probability distribution according to the measured toxicity while decoding.

Post Filtering Instead of encouraging a neural conversation model to stick to the social norm, this intervention point prevents the model from inappropriate contributions by denying already generated contributions. Dinan et al. (2019a) introduced a four-step approach to train a robust detection system: Build an initial model on the Wikipedia Toxic Comments dataset that detects single-turn utterances. Then ask crowd workers to find contributions that are not covered by the model. Retrain the classifier and repeat the crowd-working task again. Since single-turn analysis is not sufficient, they also experimented with crowdsourced multi-turn samples and reached an F1 score of 66.4 on their best multi-turn classifier. Xu et al. (2020) followed up on this work, but focused on eliciting offensive messages from the neural conversation model in the break-it phases. For that, they formulated four different strategies: training classifiers for detecting unsafe messages, making unsafe content explicitly unlikely, avoiding topics like politics or religion and forcing the model to respond gender-neutral. They reported an F1 of 85.9 on the same multi-turn data, and humans rated their best model as safe in 96,6% of the time.

3.5. Plan for the Rest of the Thesis

Contributing to a conversation cannot be solved by only feeding large dialogue corpora into generative neural networks, as elaborated in this chapter. That only makes models fluent. Informativity can be achieved by controlling the token diversity and a lot of research in that field yielded good output already (though we think that the core issue of informativity, the model learning process itself, is not solved). In this thesis, we want to investigate into consistency. We bring together the fact-based and opinionated consistency by creating a new dataset: KOMODIS in Chapter 4. We explain the collection process, analyze the dataset itself and conduct a series of experiments on standard neural conversation models. Then, in the following chapters, we experiment with different approaches to specifically improve the consistency of neural conversation models using the new dataset: Models that use additional context have the issue of limited input data size. Therefore, we investigate a new concise encoding for Transformer-based models in Chapter 5 and evaluate our approach regarding space efficiency and knowledgeability. In Chapter 6 we validate suitability of reinforcement learning instead of standard language modelling and in Chapter 7 we combine multiple dialogue datasets to see the influence of transfer learning approaches on consistency.

4. Context-augmented Data for Dialogue Generation

This chapter is based on our publication: A Corpus of Controlled Opinionated and Knowledgeable Movie Discussions for Training Neural Conversation Models (Galetzka et al., 2020). I was responsible for collecting the dataset, conducting the experiments and writing the publication. Some content from it appears verbatim here and will not be specifically credited in this thesis.

Previous research focused on either consistency towards facts or personalities, and in this chapter we aim to investigate the combination of both. If a conversation system produces only consistent utterances, it should be globally coherent (as long as the generated content relates exclusively to the facts and opinions available). Figure 4.1 illustrates global incoherence in these two dimensions, even though each utterance from the system is locally coherent (a good reply to its immediate precursor). Therefore, we plan, collect and validate a new dataset that consists of natural human-human dialogues in the domain of movies. Every dialogue is augmented by a set of facts and opinions that are utilized by the dialogue partners within each conversation.

We first define requirements for the dataset in Section 4.1, describe the technical setup in Section 4.2 and the data collection in Section 4.3. The final dataset is described and validated in Section 4.4. Finally, in Section 4.5 we enhance the dataset for NLI problems.

4.1. Requirements

We aim to create a dataset with dialogues based on a set of given facts and opinions. The dataset should serve as a fine-tuning dataset for language models and should enable research for consistency (as defined in chapter 3) in neural conversation models. To achieve that, we define the following requirements in no specific order:

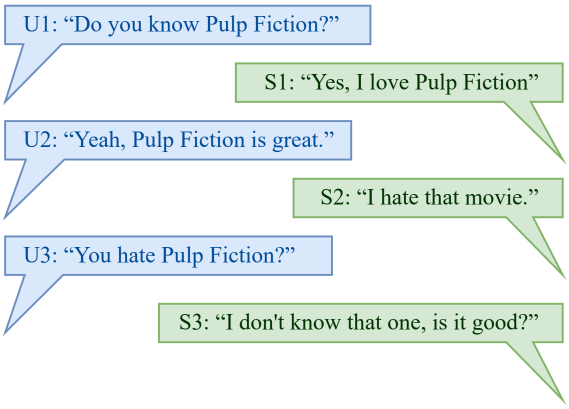

Figure 4.1.: A constructed example of a dialogue between a user (in blue) and a conversation model (in green) that is locally coherent, but globally incoherent along the dimensions knowledge (S3 to S1, S2) and opinion (S2 to S1).

Fine-tuning Capabilities Usually, state-of-the-art neural conversation models are built by fine-tuning a pre-trained language model on specific dialogue data. As we aim to create a dataset for the latter, we weighted quality higher than quantity. Table 4.1 lists number of dialogues and utterances of two similar datasets. Further details on these datasets can be found in Section 2.3.2. Our dataset size should be designed similar to them. However, both datasets are open-domain and (as explained in the next paragraph) we plan a closed-domain dataset, so the required data size decreases. Therefore, we set the minimum size of the dataset to 5,000 dialogues.

Good Generalization Language models for neural conversation models are pre-trained on big plain text sources. Therefore, utilization of additional dialogue context has to be learned from scratch at fine-tuning. This requires good generalization at training and since the data source (quality and quantity) is an important factor for generalization, we have to make sure that the dataset is sufficiently good for that. Although an open-domain setup would fit the idea of non-goal driven conversations more, we decided to stick to a closed-domain. With this decision we avoid the risk of bad generalization through a data bottleneck and non-goal driven conversation can be restricted by a closed domain without major downsides. However, while interacting with these models, the users have to stay in that domain throughout the conversation.

Natural Dialogues Dialogues can be created in different ways. The number of participants per dialogue can be one (creating self dialogues), two (a real dialogue) or more (any person

dataset	number of dialogues	number of utterances
PersonaChat (Zhang et al., 2018)	8,939	162,064
Wizard of Wikipedia (Dinan et al., 2019c)	22,311	201,999

Table 4.1.: Dataset sizes of similar crowdsourced datasets. PersonaChat consists of dialogues augmented by a personalty profile and Wizard of Wikipedia contains dialogues with knowledge support from Wikipedia articles.

can create the next contribution). Number of total participants, and therefore number of contributions per participant, is important as well. Our goal is to collect interactions that are as natural and broad as possible. However, we did not consider collecting spoken dialogues, as most of the research focuses on written language and a spoken dialogue setup would require more resources for collecting and transcription afterwards. To ensure naturalness we have to reduce the restrictions for utterance length and number of speaker turns to a minimum. Further, the dialogue partners should be able to chat in real time and number of different participants should be maximized.

Focus on Facts and Opinions One important part of our dataset is the augmented context per dialogue. With our research we want to combine consistency regarding facts and consistency towards opinions. Together with the previous requirements we decided to set our domain to *movies*. This domain offers potential for opinionated discussions, factual knowledge and is generally an engaging topic for many people (as we assume). We found that structured knowledge about movies is available on the internet, and therefore it is possible to create knowledge graphs and augment them by opinions towards entities (like actors, movies or genres).

We realize the data collection with the crowdsourcing platform AMT [1] (see section 4.2.1). Since this platform does not support multi-user interaction by itself, we use the interaction server tool slurk (Schlangen et al., 2018) (see section 4.2.2) to connect participants from AMT into different chat rooms. The technical setup is explained in the next section 4.2.

[1] https://www.mturk.com/

	requirements	realization
1	Should consist of natural human-like dialogues.	Real-time chat setup with Amazon Mechanical Turk (see section 4.2.1) and slurk (Schlangen et al., 2018) (see section 4.2.2).
2	Should serve as a fine-tuning dataset for language models.	Should consist of at least 5,000 dialogues. Number estimated by observing similar datasets.
3	Should generalize well across dialogues, facts and opinions.	Closed-domain dataset, as it is expected that generalization across any topic requires more data than we set in requirement 2.
4	Should cover facts and opinions.	Dataset is set in the movie domain, as there are huge databases available (see section 4.2.3) and movies are a common topic for opinionated discussions.

Table 4.2.: Requirements and realization strategies for the KOMODIS dataset.

4.2. Technical Setup

In this section, we describe the technical setup used to collect the dataset. We combine the crowdsourcing platform AMT and the interaction server slurk to create real time chatrooms for our participants. The general concept idea is as follows: crowd-worker from AMT that choose to work through our task are redirected to a virtual waiting room on our own server. Each crowd-worker has a unique identification number and name. The virtual waiting room pairs up crowd-worker and sends them into separate chatrooms. Each chatroom has a unique identification number and provides context (about movies) to each participant. After they chatted with each other, a unique task token is created and the crowd-worker are redirected back to AMT. Now they have to finish their task by submitting that token. Later, this token is used to identify the chat partner and pay them according to their contribution to the dataset collection. The setup is visualized in Figure 4.2.

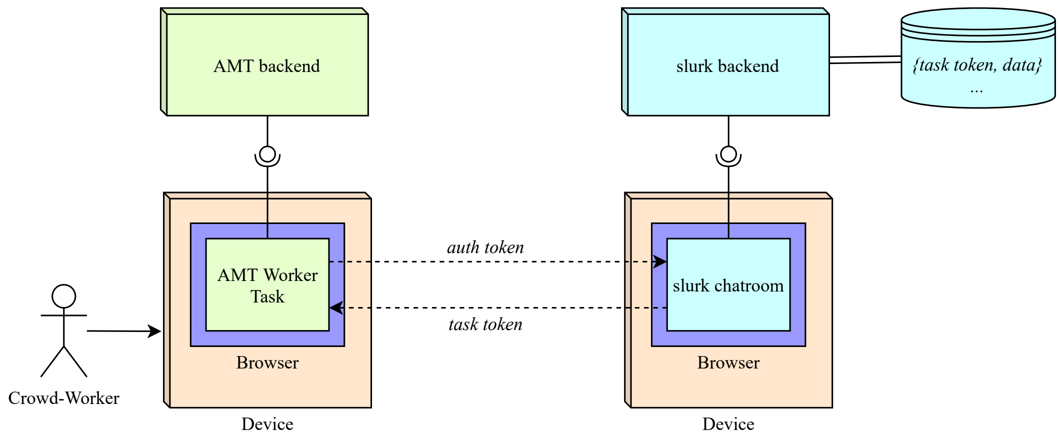

Figure 4.2.: Overview of our technical setup for dataset collection. Crowd-worker are redirected from AMT to our interaction server. After they have chatted with another crowd worker, they get a token and are redirected back to AMT. We track all interactions on our interaction server.

4.2.1. Amazon Mechanical Turk

AMT [2] is a crowdsourcing platform where people can create tasks (called Human Intelligence Task (HIT)) that are processed by crowd-workers. This section focuses on the technical implementation. Data collection design is described in 4.3.

Usually, a HIT is designed to be done within the AMT interface and server. Requester can provide an HTML design according to their task. For each sample of the task (e.g., labeling a picture) a unique HIT is created. All necessary information (e.g., a link to the picture) is transmitted to AMT for every HIT. Crowd worker can now preview and accept these HITs via the AMT server. After acceptance, they have a limited amount of time to process the HIT through the provided graphical user interface. Actions by the crowd worker (e.g., writing a label for the picture) are stored on the AMT server and can be viewed afterwards by the creators.

Since we collect the data on our interaction server, the HTML design for a HIT is simplistic. We have to provide a link that redirects the crowd worker to our server, a text field for the final token and a submit button to finish the task. The design is shown in Figure 4.3. Instruction slides and task evaluation metrics for the crowd-worker are explained in section 4.3.

[2] https://www.mturk.com/

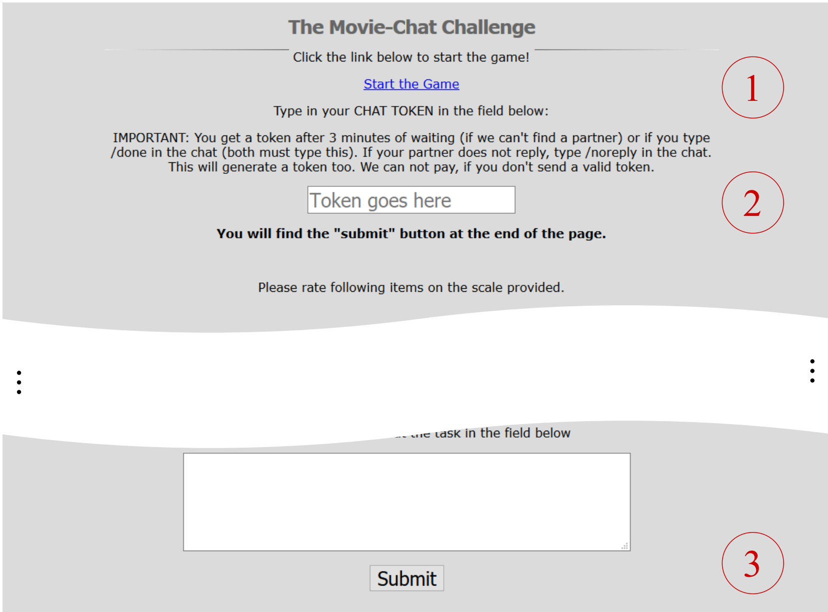

Figure 4.3.: Graphical user interface for our HIT after instructions. Crowd worker have to click the link "Start the Game" (1) first to get redirected to our slurk server to participate in our dialogue generation task. After completion, crowd-worker have to submit the generated token into the HIT interface (2) and press submit (3) to finish it.

4.2.2. Interaction Server for Dialogues

To collect the dialogues, we use the lightweight interaction server slurk [3] (Schlangen et al., 2018). The tool is a chat server and web framework (implemented in Python and Flask [4]) that allows users (clients) to connect via web browser to chat rooms. The graphical user interface contains a chat history and an area for additional information (like images or—in our case—movie facts and opinions).

We set up one waiting room that clients can connect to asynchronously. Chat functions are disabled and if two clients are in that waiting room at the same time, a new chat room instance is created for them. Additionally, a chat master bot instance is added to that room as well. The purpose of that bot is to post instructions and to react to specific client input as a third person in that chat room. More details are given in section 4.3. For each chat room instance a unique set of movie facts and opinions (see next section) is displayed to the clients. All interactions are logged to create the data set afterwards. We show a brief overview of this setup in Figure 4.4.

[3] https://github.com/clp-research/slurk
[4] https://github.com/pallets/flask/

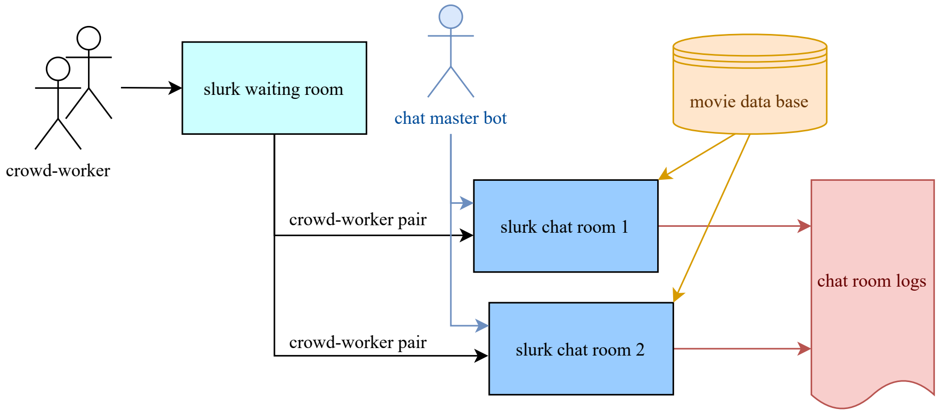

Figure 4.4.: Overview of the dialogue collection task in slurk. Crowd-worker join the waiting room asynchronously. If a pair of crowd-worker are available, a new chat room instance is created and the crowd-worker and a chat master bot are assigned to that room. A set of movie data is added to the chat room as well. Every chat interaction is stored in a log file. After the task the chat master bot creates tokens for both participants.

After the users agree to finish the chat, a unique and private token is sent by the chat master bot. These token need to be submitted in the HIT and are later used to match AMT users and chat clients. This is necessary to pay them appropriately and deny payment if they violated our rules (e.g., not chatting at all). If clients wait in the waiting room for too long (we set this threshold to 3 minutes) they also receive a token to finish the AMT task. We pay them an appropriate amount for the waiting time. We list more details in Section 4.3.

4.2.3. Movie Database

To create the movie facts, we scraped the Internet Movie Database (IMDb),[5] an online database for movies, series and TV shows. For the API we used the Python package IMDbPy.[6] The database consists of many movies that are not valid for our purpose, such as very unknown movies, movies without or with fewer facts, non-English movies and movies with unacceptable genres (e.g., pornographic films). Movies and actors have a list of what they call *trivia*. These are short facts about them which can be added and up voted by the IMDb community. We chose 500 movies based on the following rules:

1. Take the 1,000 most popular movies (popularity is given by IMDb).

[5] `https://www.imdb.com`
[6] `https://imdbpy.github.io/`

fact type	**content**
movie title	Pulp Fiction
plot	The lives of two mob hitmen, a boxer, a gangster and his wife, and a pair of diner bandits intertwine in four tales of violence and redemption.
release year	1994
genres	crime, drama
age certificate	16
budget	$8,000,000
actors	John Travolta, Uma Thurman, Bruce Willis, ...
director	Quentin Tarantino
writer	Quentin Tarantino
trivia	1.) Bruce Willis worked on the film for only eighteen days. 2.) Quentin Tarantino was quoted as saying that Butch is responsible for keying Vincent's car. 3.) ...

Table 4.3.: Information scraped from IMDb for the movie Pulp Fiction. Actors and trivia list is incomplete.

2. Add the 250 best-rated movies (provided by IMDb as a special list).
3. Filter movies that do not have a short plot.
4. Take 500 movies with the highest number of movie and actor trivia.
5. Go through the list manually to check if all movies are appropriate.

The list of movies was created in 2018 and since the database is dynamic, it is not possible to reproduce the list. It can be found in the according repository. For each movie we scraped the ten most popular actors, the director and writer, a short plot, genres, age certificate, release year and budget. Additionally, we scraped trivia about the movie and all persons. Since there are many trivia, we used the following rules for each entity:

1. Sort trivia by number of up votes.
2. Filter out all trivia below a 50% threshold.
3. Sort trivia by length (the shortest trivia is highest).

These rules ensure the use of short trivia that have a high number of up votes. From careful observations we found that these trivia are well-written and contain crisp facts that can easily be used in conversations. We store all data in a local database. In section 4.3 we explain the strategies to create fact and personality profiles out of this database. As an example we list all scraped facts for the movie *Pulp Fiction* in Table 4.3.

4.3. Data Collection

In this section, we describe the data collection process. We first give details on how we generated fact and personality profiles for the participants in section 4.3.1, then describe the collection task with instructions, automated quality checks and payment strategies in section 4.3.2 and lastly post-processing the collected data to the actual dataset in section 4.3.3.

We aim to create dialogues about one specific movie. Each dialogue partner should see some information about the movie. We want to cover all cases: both partners know similar information, both partners know different information and one partner does not know much about the movie. Further, we want to engage the partners to ask questions. Additionally, we want to provide opinions towards these facts. Beside the constraint to use and follow the facts and opinion profile, we would rather not restrict the conversation.

4.3.1. Fact and Personality Profiles

For the dialogue collection we create unique pairs of speaker profiles P_A and P_B for each dialogue. To induce interesting dialogues, we varied the relations between P_A and P_B. Some pairs have the same knowledge facts and opinions and therefore are identical:

$$P_A = P_B \tag{4.1}$$

Some pairs are complimentary but not conflicting. For example, P_A knows something that P_B does not and vice versa. Formally, the feature structures representing the profiles can be unified:

$$P_A \sqcup P_B \tag{4.2}$$

Additionally, pairs can be incompatible along opinions, which should induce opinionated discussions:

$$P_A \sqcup P_B = \bot \tag{4.3}$$

We do not include incompatible facts as this would require us to add wrong facts, and we do not want the dialogue partners to get into factual discussions. Figure 4.5 shows a generic speaker profile pair. We create these profile pairs by iteratively sampling facts from our local database (see section 4.2.3) and augmenting the occurring entities by additional opinions. To ensure the above relations between profiles, we developed a generation algorithm with specific rules.

Questions To get the participants to ask specific questions that the dialogue partner can answer requires additional effort on the speaker profiles. We decided randomly to add the task to ask a specific question (e.g., "*At some point in your chat, please ask for the age restriction of the movie!*") to one speaker profile and make sure that the other speaker profile does contain the answer to it.

Unknown Movies We also consider profile pairs where one participant does not know the movie at all. This requires a specific setup, since these participants must not see any facts about that movie in their profiles. For these profile pairs we add at least one question (as described in the previous paragraph).

Facts If both P_A and P_B are supposed to know the movie, we provide 2–4 facts with respect to the relations defined above. Facts from trivia sentences are related to one or more entities (e.g., the trivia from Table 4.3 is about Bruce Willis and Pulp Fiction) and to ensure a fluent speaker profile pair we make sure that occurred entities (by randomly sampling a fact) are re-used for the following facts. For example, if the first randomly sampled fact states that Bruce Willis is an actor of Pulp Fiction, the algorithm tries to choose a trivia about Bruce Willis next.

Trivia Trivia are handles as open domain facts about movie related entities. Since they are freely written and our database contain many of them, we decided not to choose a trivia twice. This increases the data diversity of our dataset and therefore generalization while training neural networks. We preprocessed all trivia with simple special character cleaning and replaced occurrences of the following regular expressions with a white space:

```
regex_list=["(qv)", "([^)(])", "[[^][]]", "_", "\s+"]
```

This removes content within brackets and characters around linked entities to have clean and crisp facts. Since we use every trivia only once, the number of speaker profile pairs is limited, especially for less famous movies as they have fewer trivia.

Opinions We create an opinion for every entity mentioned in the speaker profile pairs. Opinions are discrete, and for every opinion-entity combination we created sentence patterns to make them easily understandable for the participants. The following discrete opinions exist (one example pattern in brackets):

Profile (Person A)	**Profile (Person B)**
Talk about <*movie*>!	Talk about <*movie*>!
Plot of / Fact about <*movie*> *	Plot of / Fact about <*movie*> *
Fact about <*other_entity*> **	Fact about <*other_entity*> **
Opinion about <*movie*>	Opinion about <*movie*>
Opinion about <*other_entity*> **	Opinion about <*other_entity*> **
Task: Ask for specific fact *	

Figure 4.5.: A generic speaker profile pair P_A, P_B for the dataset generation. * denotes information that may not occur in every profile. ** denotes the possibility of multiple occurrences with different entities.

1. Favorite (”*I think <movie> is my favorite movie.*”)
2. Strong like (“*<person> is a great actor!*”)
3. Like (“*I like <genre> movies.*”)
4. Dislike (“*I don’t like <person>.*”)
5. Strong dislike (“*I really don’t like <movie>.*”)
6. Unknown (“*I don’t know <actor>.*”)

We decided to decrease the sampling probability of extreme opinions (like favorite) to better represent natural dialogues. Diagrams in Figure 4.5 show the number of occurrences for each opinion across all generated persona profile pairs.

With the above restrictions and strategies we were able to generate 11,497 unique speaker profile pairs across 500 different movies. It would have been possible to create more, but that would cause an imbalance in pairs per movie, as trivia for less popular movies are limited. And since one requirement is to generate at least 5,000 dialogues, we stopped processing more.

4.3.2. Collection Setup and Constraints

To ensure that crowd-worker create suitable dialogues, we have to make sure that the task is understandable and as simple as possible, that crowd worker get paid appropriately and that there is sufficient protection against scam. In order to instruct the crowd worker, we created five instruction slides that are shown before the worker can enter the waiting room on slurk. The original slides can be found in Appendix A.

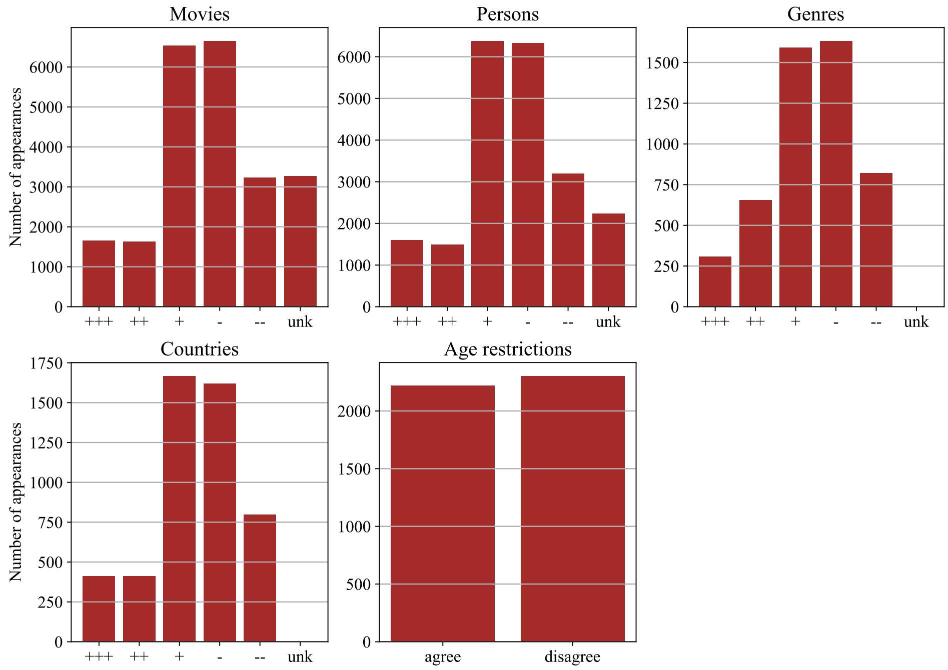

Figure 4.6.: Number of opinion appearances for different entities on all generated profiles. Discrete opinions range from favorite (+++) to strong dislike (- -) and unknown.

The first slide describes the task in general and is designed in a way that it imitates the chat rooms (chat on the left and speaker profile on the right). We give a brief overview first ("Your task: Conducting a conversation with another person about movies.") and then explain the (less intuitive) speaker profile in greater detail. As explained in section 4.3.1 we differentiate between facts and opinions. Additionally, we create separate tables for questions and facts that should only be used if the dialogue partner asked for.

Payment AMT provides a payment system that differentiates between basic and bonus payment. The basic payment is bound to the HIT and needs to be specified beforehand. A bonus payment of any value can be granted after the HIT was successfully done. We make use of both payments, and the second introduction slide explains that to the participants. We set the basic payment to 0.10$ and grand this for waiting in the waiting room. If no dialogue partner is found within 3 minutes, a token is generated and the participant can finish the achit. 0.10$ for 3 minutes is a low hourly wage, but it is granted for doing nothing and should not be the default case any way. If the participant is paired up we pay an additional 0.50$ for a 3-minute chat which we call a bonus. A participant can always end the dialogue by typing

"/done" in the chat, but if the chat lasts less than 3 minutes, the bonus is reduced accordingly. To further engage participants, we provide a super bonus of 0.20$ for "*at least 7 sensible, well-written speaker turns*." The introduction slide 3 explains that to the participants. While we track speaker turns, we do not test for utterance quality automatically. That means in our process, the super bonus is not bound to well-written utterances at all. However, we do evaluate the quality of participants by asking their chat partners afterwards (see next paragraph).

All together, we come to a payment of up to 0.80$ for a 3-minute chat. Since the task (dialogue) is dependent on two participants, it sometimes creates issues for fair payment. Whenever one participant is careful but the other one responds slow or not at all, even the careful participant cannot get the full bonus. By tracking response times and automatically evaluating them, we make sure that participants get paid well, even if we do not have a good dialogue at the end.

Dialogue Quality and Scam Tracking dialogue quality automatically is not precise enough to bound payment to it. Due to the required dataset size, it is also not possible to track quality by ourselves. Therefore, we ask the participants after the task to rate their dialogue partner. We provide a 5 point Likert scale from *strongly disagree* to *strongly agree* for the questions:

1. My partner chatted like an attentive person would.
2. My partner is a native speaker of English.
3. This felt like a natural chat about movies.

We don't use these ratings for bonus payment estimation, but we block participants with bad ratings for future dialogues. We also block participants that did not respond at all after matching them or were unable to use the "*/done*" command correctly at the end of the conversation.
AMT provides qualification requirements that can be specified for a HIT. We do add the following constraints to reduce scamming, non-native participants and general quality:

1. Participant location has to be one of: Great Britain, United States, Australia, Canada or Ireland to reduce non-native participants.
2. Number of accepted HITs has to be greater than 2,500 to reduce new accounts that are supposed to scam HITs.
3. Acceptance rate of HITs has to be greater than 0.95 % to reduce accounts that are supposed to scam or with generally weak performance on tasks.

These qualification requirements are commonly set in similar ways for other tasks in the literature. Even though this greatly reduces scamming and bad quality, there are still participants that try to scam, are non-native, do not understand the task or write inappropriate utterances. We did some tests before collecting the real data and noticed some issues, which we will describe in the following paragraphs.
Slide 4 is supposed to give additional hints to the participants so that they understand what they need to do. An additional issue that occurred was that participants started to chat about the HIT itself, especially after they were done. Common phrases we found are: "*Are we done with the chat?*" or "*How do I get the token now?*". Such utterances are difficult to filter out automatically in the post-processing step, so we had to explicitly mention that.

Cross Talk In a spoken conversation between two or more dialogue partners, only one person can speak at the same time, since the other participant needs to listen to the speaker. In a chat conversation, however, the participants have to write what they want to communicate first. The participants can write at the same time, and you can read your partner's contribution only after they sent it into the chat. This can cause cross talk, which means that a dialogue partner could answer to the penultimate utterance rather than the last utterance, as you would in a spoken conversation.
Cross talk can be tackled by enforcing strict turn taking throughout the interaction. This, however, slows down the conversation and can reduce naturalness. Further, if participants press *enter* too early (sending their reply in the chat), they cannot fix that by adding something else, which happens frequently in a chat. We noticed that most cross talking happens at the beginning of the conversation, as both participants start to write at the same time. We therefore randomly choose one participant and task him to start the conversation, while blocking the chat window for the other one for a few seconds or until the first message was sent. This greatly reduces cross talking. Additionally, we added an *is typing* feature to give feedback if the chat partner is currently writing. This is a common feature in modern chat programs anyway and also helped to reduce cross talking as participants are more patient if they see that the partner is interacting.

Copy-pasting Speaker Profile Information In our early tests we noticed that participants tend to copy information from the speaker profiles. To tackle that, we disabled text selection on the speaker profiles within slurk. This reduced exact copy-pasting of facts or opinions, but did not hold them from writing it off manually. However, these utterances were grammatically correct, as they put more effort in making a good reply. In numerous instances, especially for the trivia facts, a good way of using them is to use parts of them verbatim. Therefore, we do not check for word overlap to force participants into rewriting.

Rushing through the Speaker Profile Related to the previous described copy issue is the problem of short unappealing dialogues. Participants tend to rush through HITs in the first place or the tasks seem to become boring after several repetitions, as we experienced a drop in dialogue diversity and quality over time. The latter can be prevented by setting a daily limit for each user account. We set the value to 5, as lower values seem to reduce the number of matches as fewer participants can do the task at the same time.
The former issue is harder to tackle. Using the participant evaluation results to not only block scammer (as described in the dialogue quality and scam paragraph), but also to filter out more participants (e.g., below average) to increase dialogue quality, causes two issues: First, if too many participants are blocked for that task, it is harder to match participants, which reduces the number of dialogues that can be collected. Second, participants tend to rate their chat partners more positively than we would, which makes it hard to find a threshold for blocking. Instead, we experimented with task length (3 minutes) and number of required speaker turns for additional bonus payment (7 speaker turns) until the results were good enough. Less time and more speaker turns lead to shorter and more unappealing dialogues, as expected.

Payment and Quality Correlation Dialogue quality seems to also correlate with the amount of payment, which we also expected. As described earlier, we decided to set a maximum of 0.80$ per participant, which results in up to 1.60$ per dialogue.

One final slide is added at the end to cover necessary information beside the dialogue task itself. Participants can write "*/done*" to signalize that they think the dialogue is sufficient. If the dialogue partner also writes that command, the task is finished and individual tokens are generated by our system. If a dialogue partner does not respond for at least 30 seconds, participants are allowed to type "*/noreply*" instead to get a token by our system to finish the task nevertheless.

4.3.3. Post-Processing

We initially collected 8,000 interactions. The first post-processing step is to filter out inappropriate conversations. 784 (9.8%) interactions had less than 3 speaker turns mostly due to

named entity resolution	
precision	0.977
recall	0.920
f1-score	0.947

Table 4.4.: Evaluation of our named entity resolution. Values were estimated manually on 50 randomly chosen dialogues that were not used to develop the algorithm.

non-responding participants. 248 (3.1%) conversations were filtered out because one partner strongly disagreed with at least one of our evaluation metrics. We blocked these user accounts mostly within 24 hours of the occurrence to keep these interactions as low as possible. A second batch produced another 551 dialogues, which results in 7,519 total dialogues after the simple filtering. We cut all utterances after the first appearance of a special command (either 'noreply' or 'done') by any participant. We found that this is an efficient and straightforward way to remove unintended interactions from the dialogue. After these commands the dialogues are either over anyway or there is a significant amount of non-movie related interaction.

Since we did not restrict the participants to strict turn-taking, utterances of the same dialogue partner can appear one after the other. The final dataset, however, should have utterances from alternating speakers. We simply concatenated these utterances and added a dot in between, if the first one does not end with a punctuation mark.

Since we know about which entities the dialogues are about, we were able to resolve named entities (movie titles, persons, release years, genres and age certificates) with very high precision and recall. We list metrics for our final algorithm in Table 4.4. Detecting movie titles is a challenging problem as the search space is high. Since we have a small number of entities to look for, simple rules are sufficient to find the entities. Most problems were caused by grammatical errors and spelling mistakes. Our algorithm to detect entities consists of four steps:

Exact String Match on lower cased strings for all entity types. This step has high precision on persons and movie titles, as if they were mentioned exactly as shown in the speaker profiles, they almost always were meant by the participants. However, the crowd worker often used shorter versions of them (e.g., *DiCaprio* instead of *Leonardo DiCaprio* or *Harry Potter* instead of *Harry Potter and the Goblet of Fire*). Obviously, this rule does not cover spelling mistakes. Genre and age certificate sometimes caused false positives because even the exact strings are generic: The genre *crime* could also be used in the plot ("A movie about *crime* and violence.”). Validation showed that this did not happen often and therefore this rule is still precise. For budget we chose all numbers greater than 1 million, with and without delimiter and considered the word *million* by using the following regular expression:

```
regex=["\$?[0-9]{1,3}[,\s]?(\d{3}[,\s]?\d{3}|\b(million)\b)"]
```

Then converted them into an integer and compared them to the real budget of the movie, if present in the speaker profile. We classified them as the budget if integer values are the same.

n-gram Comparisons for movie titles. Since many participants used short versions of the movie names, especially when the movie names are longer, we compared every n-gram in an utterance and the movie title with cosine similarity, if the movie title consists of at least 3 tokens:

$$\min(t_{movie};3) \leq n \leq t_{movie} \tag{4.4}$$

with t_{movie} as the number of tokens of a movie. We classified an n-gram as the movie title if the cosine similarity is equal to 0.9 or higher. We implemented this rule in Spacy (Honnibal and Montani, 2017). This rule has high precision but does not find all occurrences, mostly because of spelling mistakes.

Jaccard and Levenshtein Distance for movie titles. The third step is a comparison of the same n-grams with a combination of the Jaccard distance (Niwattanakul et al., 2013) with threshold ≤ 3 and Levenshtein distance (Levenshtein, 1966) with threshold ≤ 0.3. This rule is more robust against spelling mistakes but cannot be used for 2-grams or single token comparisons, as it would require us to lower the thresholds (a spelling mistake on a short movie title increases the distance by a higher amount compared to long titles) which causes false positives on some movies.

Named Entity Recognition from Spacy for persons. We found that the internal model for person recognition in Spacy works well on movie actors, directors and writers in our dataset. Therefore, in addition to the exact string match, we added all persons detected by Spacy where at least one name (surname or first names) matches one from the speaker profile. This rule does not cover spelling mistakes, but after extensive investigation we found that names were mostly spelled correctly, even upper-cased.

We use the results of the named entity resolution to provide two versions of our dataset. One version with the original dialogues and one with substituted named entities. The latter is used for data quality validation (see section 4.4.3) and enables research experiments where it is necessary or important to have detailed labels for the entities. Further, these annotations allow cleaning the original data by using the correct movie titles and full names instead of the ones written by the participants. Both versions should decrease the generalization effort a neural network has to make to produce utterances that are consistent to the speaker profiles. Though the ultimate goal should be to learn on the original data, and thus we think that these simplified versions open up pre-training, bootstrapping or evaluation possibilities to reach that goal.

4.4. KOMODIS Dataset

In the following section, we present and discuss our final dataset that we call KOMODIS. We first give a general description and show a full example in subsection 4.4.1. We do a quantitative analysis of the data in subsection 4.4.2 and validate the data quality in subsection 4.4.3. Finally, we apply the dataset on a pre-trained Transformer-based model to validate its possibility to serve as a conversation dataset in subsection 4.4.4.

4.4.1. General Description

The dataset consists of 7,519 dialogues across 500 different movies with a total of 103,500 utterances, which infers an average number of 13.8 utterances per dialogue. More statistics are discussed in the next section. Every sample is a conversation about one specific movie between two speakers. Each is augmented by two speaker profiles that consist of facts and opinions related to that movie. Entities in the dialogue are linked to the according entities in the speaker profiles. We show one example dialogue with speaker profiles in full length and with original spelling in Figure 4.7.

Like many others, this dialogue starts with a brief greeting. One downside of our domain restricted setting is, that many dialogues start similar. Someone watched a movie or want to watch a movie is quite a common scenario. Participants often used all facts and opinions at some point in the conversation, like in the example. Punctuation also differs. In this case, speaker *A* always starts with a capital letter and ends all sentences with a punctuation mark, while speaker *B* did not. Most dialogues follow the speaker profile with no or less small talk about other things, but the trivia facts. However, the conversations itself have small talk characteristics and therefore are non-goal driven, as required.

For each dialogue facts and opinions for the speaker profiles are encoded as triples and for the opinions full sentences are added in addition to them. Dialogue and speaker profiles are stored with and without named entity replacements. A named entity dictionary is added to resolve named entities. We show an example for speaker A and a part of the dialogue in Figure 4.8. Instead of speaker A and B the dataset refers to first and second speaker. The first speaker always starts the conversation (as the name suggests) and then the speaker alternate for each utterance.

4.4.2. Dataset Statistics

In this section, we present and discuss the dataset statistics of KOMODIS. It consists of 7,519 dialogues, which we classify as medium-sized. It has fewer dialogues than the PersonaChat dataset (8,939 dialogues), but since our dataset is set in a closed domain the amount of data is sufficient. The dataset has 130,500 utterances and therefore 13.8 utterances per dialogue. With 1,487,284 tokens in total, each utterance has an average of 14.4 tokens. That means the dialogues are relatively rich and long. We summarize the statistics in Table 4.5.

The total vocabulary size of the dialogues is 27,658 which is unexpectedly high for a closed domain. 99% of the most frequent tokens contain 13,727 unique tokens. There is a high number of 14,530 different tokens that only occur once in the dataset. Most of them are words with spelling mistakes (e.g., ‘abonded’, ‘abouts’, ‘witth’), some are connected words (e.g., ‘pre-2000’, ‘month-long’) and a few that we were unable to understand (e.g., ‘burnuuh’). We compared our token distribution against two similar datasets in Figure 4.9. Token indices (sorted by frequency) are plotted on the x-axis, number of occurrences are plotted on the y-axis. The token distribution looks similar to PersonaChat and OpenDialKG. PersonaChat has a more flat distribution of medium-frequent words, but since our dataset is in a closed domain, we expected that. The OpenDialKG dataset has a much steeper distribution, so we conclude that our dataset has a common token frequency distribution. However, both datasets don’t have that large amount of single occurring tokens. Since all datasets were crowdsourced, we assume that these datasets were either post-processed for spelling mistakes or participants had better instructions to write clean in the collection process.

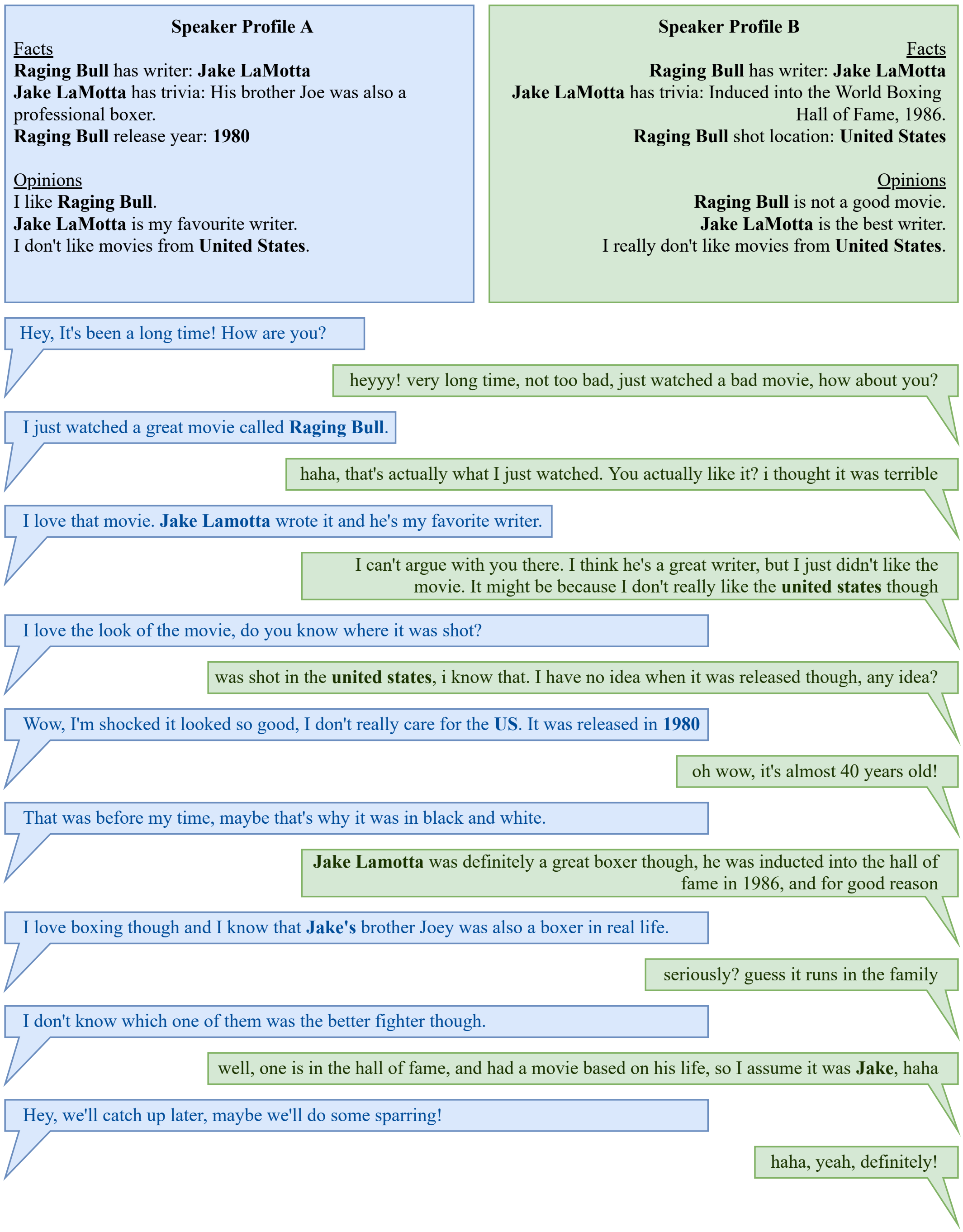

Figure 4.7.: One example dialogue from our KOMODIS dataset about the movie Raging Bull. Speaker profiles differ in terms of movie facts and the trivia about the writer Jake LaMotta. They have opposite opinions about the movie, but share similar opinions about the writer. Entities resolved by our post-processing are marked in bold.

```
facts = {
    first_speaker: [{
        subject: 'Raging Bull',
        subject_type: 'movie#0',
        relation: 'has_writer',
        object: 'Jake LaMotta',
        object_type: 'writer#0'
    }, {...}]
    second_speaker: [{...},...]
}
opinions = {
    first_speaker: [{
        subject: 'Raging Bull',
        subject_type: 'movie#0',
        relation: 'has_general_bot_attitude',
        object: '3',
        src_original: 'I like Raging Bull.',
        src_delex: 'I like movie#0.'
    }, {...}]
    second_speaker: [{...},...]
}
```

```
named_entity_dict = {
    Raging Bull: 'movie#0',
    JakeLaMotta: 'writer#0',
    ...
}
dialogue_original = [
    'Hey, It's been a long time! ...',
    'heyyy! very long time ...',
    'I just watched ... movie Raging Bull.',
    ...
    'haha, yeah, definitely!'
]
dialogue_ner = [
    'Hey, It's been a long time! ...',
    'heyyy! very long time ...',
    'I just watched ... movie movie#0.',
    ...
    'haha, yeah, definitely!'
]
```

Figure 4.8.: Part of the dialogue from Figure 4.7 encoded in the JSON format. Facts and opinions are sorted by first and second speaker and both versions (with and without named entity resolution) are available.

Each dialogue is based on one of 500 different movies. The dataset contains unique 20,489 facts and 16,065 unique opinions towards entities. That means each dialogue has on average 2.7 unique facts and 2.1 unique opinions about entities occurring in these facts. We count facts and opinions as unique, if at least one part of that fact or opinion (subject, relation or object) differs. With a total of 54,311 resolved entities in the dialogues, each dialogue contains 7.2 occurrences of entities on average. Having one entity in every other utterance means that the dialogues are indeed mostly about the movie entities, but still contain content about and around them.

569 different dialogue partners (AMT participants) were involved in generating the conversations, so on average, one participant was involved in 13.2 dialogues. We don't have data to compare with, but we think that this number is low enough so that it does not cause biases for dialogue or language behavior. Therefore, when we split the data into train, validation and test sets we did not consider a strict split of participants. It is more important to have a strict differentiation between movies. Therefore, we decided to separate the dialogues so that no movie occurs in more than one subset.

KOMODIS dataset statistics	
dialogues	7,519
utterances	130,500
tokens	1,487,284
average utterances per dialogue	13.8
average tokens per utterance	14.4
vocabulary size	27,658
vocabulary size (99% of tokens)	13,727
different movies	500
used (unique) trivia	13,818
number of facts	20,489
number of opinions	16,065
number of entity mentions	54,311
AMT participants	569

Table 4.5.: Dataset statistics for our KOMODIS dataset. The vocabulary size was computed by counting all different tokens from the original dialogues. This value varies for different tokenization algorithms.

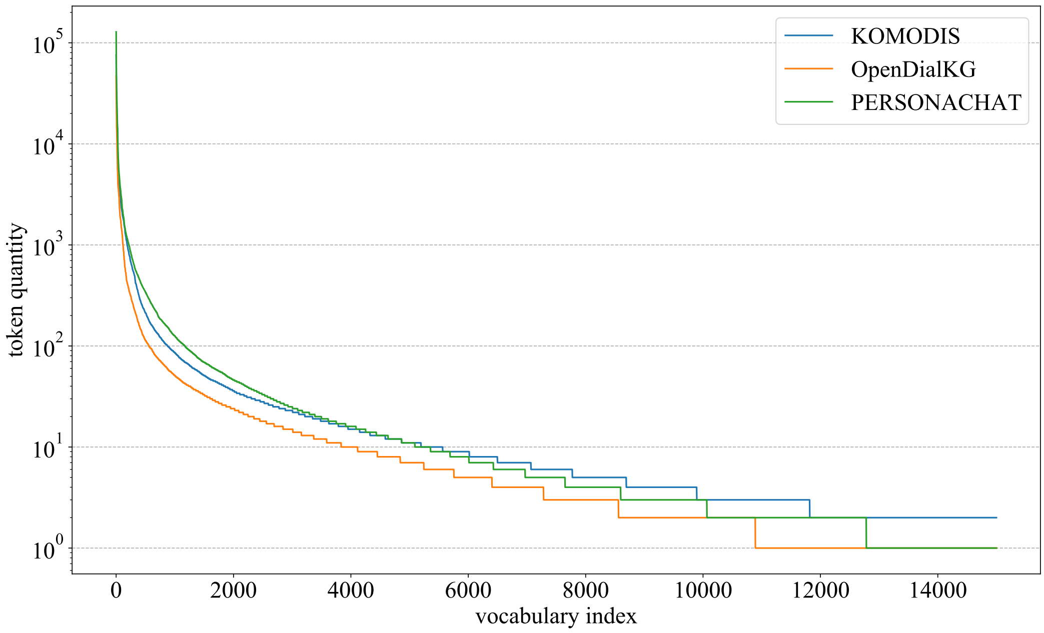

Figure 4.9.: Frequency distribution of word tokens for the KOMODIS dataset (blue) in comparison with the open-domain datasets PERSONACHAT (green) and OpenDialKG (orange).

entity	matches	errors	neutral	accuracy
movie	122	5	36	0.96
person	91	1	35	0.99
other	57	2	28	0.97
sum	270	8	99	0.97

Table 4.6.: Accuracy of participant adherence to their speaker profile. Manually evaluated on 50 randomly chosen dialogues. 'other' contains release year, genre, age certificate, shot location and budget.

4.4.3. Data Quality Validation

In addition to the analysis of the statistics in the previous section, we validate the data quality by analyzing the usage of speaker profiles. We analyze if participants really talked about the given entities and if the crowd-worker adhered to the given opinions and used the facts correctly. This is a necessary step to ensure data quality, as the reliability and suitability of the participants were not completely under our control.

To show that the participants really talked about the given facts and used the correct facts (e.g., answering with the right release year), we computed the overall coverage of named entities. For each dialogue, we compared entities that appear in the dialogue with entities that appear in the speaker profiles and computed an overlap of 93.1%. Assuming that participants did not write wrong and right facts at the same time (e.g., "The movie was released in 1990" and later: "The release year is 2005") we conclude that with a coverage of 93.1% the participants used most of the facts and used them correctly in each dialogue. Since we explicitly instructed them to not use all facts at all times if it feels unnatural in the conversation, the coverage seems appropriate.

Automated validation of the correct usage of opinions is challenging, as it requires an automated co-reference resolution and sentiment analysis tailored to our specific data. We do label all entities in terms of opinion automatically, but the quality of the results was not high enough for this validation purpose. We will explain and elaborate this further in section 4.5. Instead, we analyzed 50 randomly chosen dialogues manually. For each dialogue we identified all entity mentions and compared the opinion expressed in the utterances with the given opinion from the speaker profile. We counted matching opinions, contradiction opinions (errors) and neutral mentions (e.g., "The movie was shot in the United States" does not contain hints about the opinion of that movie.) of the entities. We list the results in Table 4.6. The accuracy of opinion adherence is 97%. We only found very few contradicting statements in

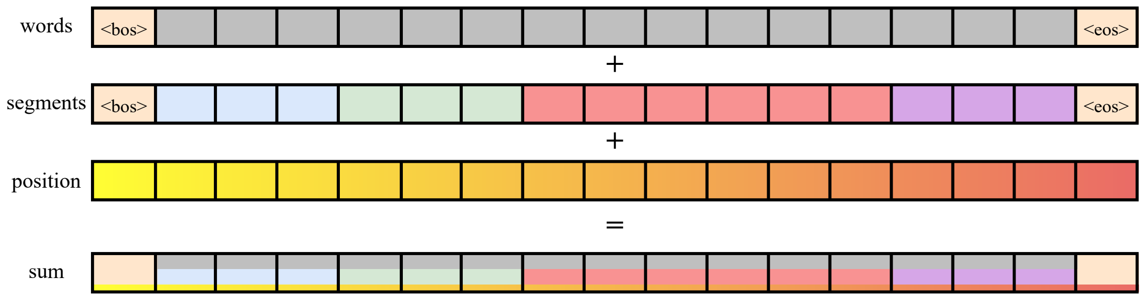

Figure 4.10.: Sequence encoding strategy as proposed by Wolf et al. (2019). Three layers of token embeddings are added to one final sequence: word embeddings, segment embeddings to differentiate between dialogue history and context and pretrained positional embeddings.

the test data. We conclude that most crowd-worker understood the task and followed their instructions across all dialogues. An error rate of approximately 3% in the training data should not cause a neural model to fail on learning opinionated utterances.

Overall, our analysis shows that the KOMODIS dataset

1. contains relatively rich and long dialogues (14 turns on average),
2. has dialogues with a primary focus on facts from the speaker profiles,
3. was created by dialogue partners that followed the speaker profiles and reproduced facts correctly.

4.4.4. Experiments

In our final validation step we train basic neural networks on the KOMODIS data to see how the dataset performs with state-of-the-art deep learning techniques. We prepare and train the dataset on the GPT-2 (Radford et al., 2018) and DialoGPT (Zhang et al., 2020) model, which are briefly discussed earlier in section 2. We first explain our different encodings, then go over the hyperparameter settings, basic decoding algorithms and finally show results on automated metrics and a human evaluation.

Sequence Encoding Both models are Transformer-Decoder architectures and therefore require one continuing sequence as input. Since the KOMODIS dialogues are augmented by a set of facts and opinions, we have to introduce an appropriate encoding so that these models can differentiate between facts, opinions and the dialogue history. Similar to Wolf et al.

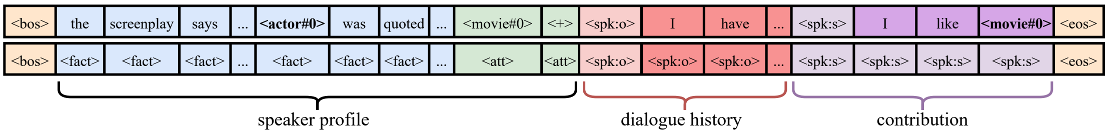

Figure 4.11.: Sequence encoding example with speaker profile (blue), dialogue history (red) and the contribution (purple) to be predicted. Word level (top row) contains the language vocabulary, and segment level (bottom row) separates the segments through new tokens defined in Table 4.7. In this example entities are replaced by generic tokens (here: actor and movie).

(2019), we concatenate speaker profiles with dialogue histories and add a new layer for context tokens (in addition to word tokens and positional embeddings) to differentiate between these tokens. Figure 4.10 shows the concept.

Figure 4.11 shows an arbitrary, more detailed example of our encoding. Facts (blue) are concatenated as full sentences in the word layer (top) and the <*fact*> token is added to the segments layer to signalize that this segment belongs to known facts. Opinions are added through abstract representations (green): In the word layer a set of tokens (from <+ + +> to <- -> for favorite to strong dislike) and the <*att*> token is added to the segments layer to signalize that this segment belongs to attitudes about known entities. In the following evaluation we refer to this abstract opinion encoding as *opinions-abstract*. Then the dialogue history is added with two segment tokens <*spk:o*> (other speaker) and <*spk:s*> as the model speaker. The last utterance is always the contribution the model has to predict and, since it's a models' utterance, it is always marked with <*spk:s*>. We also added the pretrained begin-of-sequence (<*bos*>) and end-of-sequence (<*eos*>) tokens to sequences. Further, all entities are replaced by a generic entity token that is used across all movies and entities. In the following evaluation we refer to this as *entities-delexicalised* experiments.

Instead of using abstract representations for opinions and generic entity tokens, we also experiment with opinion sentences and the original entity names, as shown in Figure 4.13. For the opinions we reuse the sentence patterns that were added to the speaker profiles for the AMT participants (green). We refer to this as *opinion-sentences* in further experiments. If we encode entities by their original names, we refer to this as *entities-original*.

Hyper Parameter Since the language model are pre-trained with specific parametrization, we use most of them by default. We list all parameters in Table 4.8. For most experiments we set the number of epochs to 3; this is discussed in the next paragraph. Parameters for the Adam optimizer are reused from pre-training. Like in the original implementation, a dropout of 10% is added for all self-attention layer at training. New token embedding weights are

context token	**description**	**initialization**
<fact>	Signalizes that word tokens belong to a fact.	random
<att>	Signalizes that word tokens belong to an opinion.	random
<spk:o>	Signalizes that word tokens belong to a user utterance.	random for GPT-2, pre-trained for Dialo-GPT
<spk:s>	Signalizes that word tokens belong to a model utterance.	random for GPT-2, pre-trained for Dialo-GPT

Table 4.7.: Description of the special tokens used in the segments layer for encoding context and dialogue.

initialized randomly with a normal distribution $N(0, 0.02)$. We keep up to 5 previous utterances as dialogue history. This number is dynamic, as the input sequence length is limited to 256 tokens, and we remove the most recent utterances if this limit is exceeded. Training data is shuffled after each epoch and we use max length padding within each batch.

Loss Function Given the model weights θ and training sequence $\mathbf{x} = (x_1, x_2, x_3, ..., x_n)$ we minimize negative log likelihood for all tokens x_i that a part of the last utterance (contribution) in that sequence:

$$\lambda_{\text{lm}} = \sum_i -\log P(x_i | x_{i-1}, x_{i-2}, ...; \theta). \tag{4.5}$$

This is a standard language modelling approach. In addition to that, we also minimize negative log likelihood of a classification layer that is supposed to classify the correct next utterance out of n candidates by:

$$\lambda_{\text{clf}} = -\log P(y | \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(n)}; \theta). \tag{4.6}$$

This retrieval-based optimization step is supposed to improve generalization and to accelerate convergence, as shown by Radford et al. (2018). Figure 4.12 visualizes this part of the model. In addition, the classification logits (hidden state that is fed to the soft-max layer) can later be used to quantify a generated sequence candidates at inference. We evaluate this later in this section.

The combined loss for training is:

$$\lambda = c_{\text{lm}} \cdot \lambda_{\text{lm}} + c_{\text{clf}} \cdot \lambda_{\text{clf}} \tag{4.7}$$

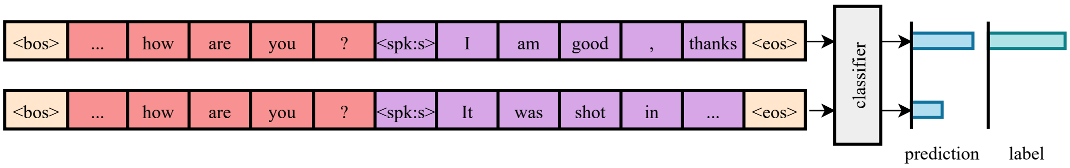

Figure 4.12.: Example of a set of $n = 2$ utterance candidates with label for the classification task of the model. The hidden states after the <*eos*> tokens are fed into a classification layer to predict the appropriate next utterance. The approach is based on the idea of Radford et al. (2018).

with scaling factors $c_{\text{lm}} = 1.0$ and $c_{\text{clf}} = 1.0$ if not further specified.

Classification Task Candidates For the classification loss λ_{clf} we have to augment each utterance of the dataset that we want to predict with a set of $n-1$ wrong utterances. In one experiment we sample these candidates randomly from the dataset (hereinafter called *random distractors*). We assume that this approach might be too easy for experiments with the original entity names (*entities-original*) as most wrong candidates will likely contain entities that are not mentioned in the speaker profiles at all. Therefore, we will experiment with random sampled utterances from other dialogues about the same movie (hereinafter called *same movie distractors*). And lastly, we want to emphasize on learning opinions and therefore randomly sample utterance with the opposite opinion of the gold utterance. This is possible due to the sentiment label post-processing of our dataset, which we will explain in section 4.5. First, we check if a gold utterance has a sentiment label. If not, we sample wrong candidates randomly. Otherwise, we estimate sentiment and entity and search of an utterance in the corpus with the same entity but the opposite opinion. If $n > 2$ we also add random utterances from the same movie in addition to the opposite opinion utterance. We will refer to this as *rule-based distractors*.

Decoding To generate utterances at inference, we use beam search with a beam size of 4. To normalize over the length of the sequences, we use the following formula from Wu et al. (2016):

$$\text{len-norm}(Y) = \frac{(5+|Y|)^{\alpha}}{(5+1)^{\alpha}} \tag{4.8}$$

with current sequence length $|Y|$ and $\alpha = 0.6$ as the length normalization coefficient. α influences the strength of the length normalization, with higher values resulting in stronger compensation, which increases the average sequence length. To avoid loops, we add sim-

default hyper parameter	
batch size	32
epochs	3
optimizer	Adam
β_1	0.9
β_2	0.999
ε	1×10^{-8}
learning rate	6.25×10^{-5}
dropout	0.1
weight decay	0.01

Table 4.8.: Default hyperparameter settings for our experiments. If one of the parameters is not specified in an experiment, this value is used.

ple beam-blocking by filtering out 3-grams that already occurred in the current or previous utterance. In addition to the accumulated beam search logits we add the logits from the classification layer to re-rank the 4 best candidates and choose the one with the highest combined score.

Epochs Automated evaluation of the validation dataset is usually used to determine when to stop a training. At some point in the training process the model starts to overfit on the train data. This can be detected by a deteriorating validation metric. Since our experiments require a lot of resources, we decided to estimate the optimal number of epochs first and then only train for a fixed number of epochs later. This is not the ideal way to find the overall optimum, but is sufficient for validating the dataset and finding a good local optimum. And, as discussed in Section 3 perplexity does not correlate with human judgement, so finding the model with the lowest perplexity should not be the most relevant objective anyway.

Thereafter, we fine-tune the pre-trained GPT and GPT-2 models on our KOMODIS data in a series of experiments to validate the appropriateness of the dataset to train neural conversation models, to create a baseline and to test the different pre-processing steps described above.

4.4.5. Results

In the following subsection, we present and discuss the results from our experiments described in the previous subsection. We reported best results on the GPT model in our paper,

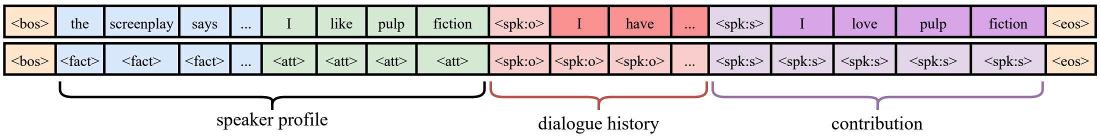

Figure 4.13.: Sequence encoding with speaker profile (blue) with full sentence opinions, dialogue history (red) and the contribution (purple) that has to be predicted. Word level (top row) contains the language vocabulary and segment level (bottom row) separates the segments through new tokens defined in Table 4.7.

since the GPT-2 experiments were done afterwards and therefore not available when we released the paper. However, it seems that GPT-2 has significantly lower perplexity compared to GPT. We list results for all experiments in Table 4.9 for GPT and in Table 4.10 for GPT-2.

Overfitting Models pre-trained on GPT and GPT-2 overfit on KOMODIS with default parameters after three epochs as shown in Figure 4.14. Both models converge fast on perplexity and then start to deteriorating in the fourth epoch. We use this as an estimation for all further experiments to stop after three epochs. The difference between GPT and GPT-2 is an offset of approximately 1 perplexity. This indicates that GPT-2 has better language modelling capacities and can learn next-token prediction on KOMODIS better. Both curves look very similar, which could be explained by the fact that both models have the same architecture and size and therefore the same information capacity. And that results in similar overfitting behavior.
As explained in the previous subsection, detecting overfitting (finding the best number of epochs) should usually be done for each individual experiment. Since we don't do extensive hyperparameter searches, but rather test different pre-processing variations, we don't expect very significant differences in our results, since optimal hyperparameter were already be estimated in the original experiments for GPT and GPT-2. We therefore decided to always stop after three epochs to save resources, which is one limiting factor in our experiments.

Opinion Encoding The opinion encoding (*opinions-original* and *opinions-abstract*) influences the perplexity slightly in all experiments. Using abstract representations increases perplexity between 0.1 and 0.7. Even though a 0.1 difference is negligible, we experienced this behavior on all experiments and especially the higher value of approximately 0.7 between GPT models with *random distractors* and *entities-delexicalised* indicates a correlation.
Abstract representations of opinions require additional train effort for the model, since we

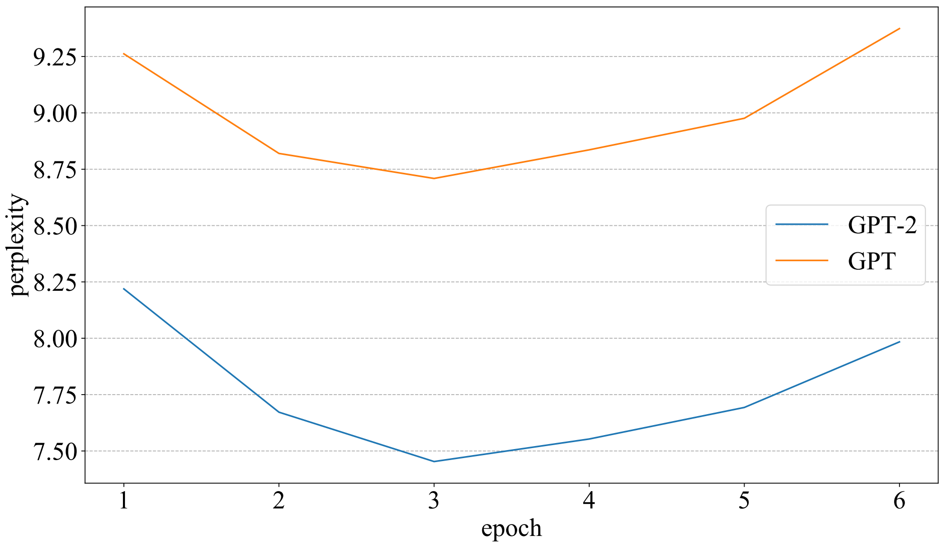

Figure 4.14.: Perplexity (lower is better) on the validation dataset for a training with default parameter. In this experiment the model starts to overfit on train data after the third epoch. GPT and GPT-2 model have similar learning characteristics, with GPT-2 having lower perplexity across all epochs.

have to introduce new (randomly initialized) word embeddings for all representations that the model has to learn. Further, by adding fully written sentences as opinions (the same that we presented the participants in our data collection process) the task to generate the gold response becomes easier as there is word overlap between the opinions and utterances. Both resulting in an overall increased perplexity on the test set. On the other hand, the small difference in perplexity shows that both models don't need plain text for the opinions and can learn abstract representations on a similar level (at least in terms of perplexity). To evaluate if the model is able to produce consistent contributions regarding opinions, we have to analyze the human evaluation.

Classification Loss For generating wrong utterances to train the classification layer, we evaluate the effectiveness of *random distractors*, *same movie distractors* and *rule-based distractors*. We evaluate this with the hits@1 and hits@3 metric, which is the percentage of gold utterances from the test set that were classified as the best candidate (for hits@3 within the 3 best candidates) out of 19 randomly sampled (wrong) candidates. GPT models trained with *rule-based distractors* achieve higher accuracy (up to 5%) compared to *random distractors*. However, GPT-2 models trained with *same movie distractors* surprisingly achieve worse results (up to 5%) compared to *random distractors*.

c_{lm}	opinions	distractors	entities	ppl	hits@1	hits@3
2	abstract	random	delex	12.94	64.44	86.61
1	abstract	random	delex	13.08	63.73	84.86
2	abstract	rule based	delex	14.65	66.27	87.28
2	sentences	random	delex	12.77	63.40	85.02
2	sentences	rule based	delex	13.94	65.89	86.82
2	sentences	random	original	**12.38**	74.22	92.41
2	sentences	rule based	original	13.50	**79.33**	**95.51**

Table 4.9.: Experiments with KOMODIS on the pre-trained GPT model. Perplexity (lower is better) is measured on the test set. Hits@n (higher is better) is measured on the test set and validates the percentage of gold utterances that are ranked within the top n utterances, with 19 additional wrong candidates.

c_{lm}	opinions	distractors	entities	ppl	hits@1	hits@1 (same movie)
2	abstract	random	delex	8.55	69.08	51.15
1	abstract	random	delex	8.60	68.70	48.10
2	abstract	same movie	delex	8.52	67.94	62.60
2	sentences	random	delex	8.30	69.85	48.85
2	sentences	same movie	delex	8.42	66.69	**66.41**
2	sentences	random	original	**8.24**	**79.01**	37.40
2	sentences	same movie	original	8.30	74.05	**66.41**

Table 4.10.: Experiments with KOMODIS on the pre-trained GPT-2 model. Perplexity (lower is better) is measured on the test set. Hits@1 (higher is better) is measured on the test set and is the accuracy of the classification layer evaluated with 19 additional wrong candidates (randomly chosen or chosen from dialogues about the same movie).

The purpose of *rule-based distractors* is to increase the difficulty at training to improve generalization at inference. The results confirm this, as the accuracy is higher. These distractors, however, increase the perplexity between 1.2 and 1.7. This does not necessarily mean less appropriate contributions for two reasons: First, perplexity does not correlate strictly with human judgement and this perplexity increase could be caused by the additional loss function. If the classification task is harder, it might increase the gradients at training (because predictions are more often wrong at training) which can lead to less effective optimization on the language modelling loss. Second, the gain in classification accuracy will help to choose the most appropriate beam at decoding, compensating the potential drop in appropriateness from the language model itself.

The idea to use *same movie distractors* is based on similar thoughts. However, our results show that this approach decreases accuracy on test set for the classification task by up to 5% on GPT-2. The perplexity increase is much smaller compared to the previous experiments (up to 0.12 perplexity). To gain more insights, we also evaluated the same models with *same movie distractors* on the test set. Here, as expected, the experiments with only candidates from the same movie produce superior results with up to 29% gain in accuracy. However, the accuracy calculated with *random distractors* is worse, but they are much closer together (a maximum of approximately 4%). We interpret this as an indication that both variants have partially independent learning goals and therefore both perform best on their own test set. This means that the *same movie distractors* characteristic is missing important parts of the *random distractor* characteristics. We can only assume that this is caused by a less variety of utterance pairs (here, a pair would be the last utterance and a possible reply, either the gold label or a distractor). And, additionally, since utterances from the set of *same movie distractors* might have less semantic variety, the probability that a supposedly wrong distractor candidate is an appropriate continuation of the dialogue is higher. That would cause some wrong candidates being appropriate instead, leading to a blurred error signal for back-propagation.

Named Entities Using the original named entities (*entities-original*) over delexicalised entities (*entities-delexicalised*) results in significantly increased accuracy (improvement of up to 15% between comparable models) for the classification loss. Using random distractors and original named entities makes the classification task easier, as the model can learn to reject next utterances, if an unknown entity (that is not mentioned in the context or history) is present. But even if trained with same movie distractors, the accuracy is higher, so there must be another reason for it. This observation requires further experiments, that we have postponed for future research. Perplexity is also slightly better for original entities. We find this surprising at first glance. We thought that a delexicalisation step should make the next-token prediction easier, since the model does not need to generalize over named entity categories. However, language models tend to overestimate the probability of frequent tokens (which causes generic utterances as described in Chapter 3) and our delexicalisation tokens are very frequent, as they replace entities in all dialogues. Using them might hurt the generation quality, causing overall worse perplexity. Additionally, the delexicalisation token embeddings need to be learned from scratch, while named entities such as names and genres already have embeddings that fit into the context of the sequences.

	naturalness	**attentiveness**	**consistency**	**personality**	**knowledge**
dataset	4.20 (0.96)	4.22 (0.91)	4.36 (0.80)	4.55 (0.72)	4.14 (0.97)
random	4.01 (0.80)	3.90 (0.93)	4.03 (0.73)	3.86 (0.65)	3.93 (0.72)
rule-based	**4.11** (0.67)	**4.09** (0.71)	**4.05** (0.56)	**4.01** (0.69)	**4.04** (0.63)

Table 4.11.: Human evaluation of our GPT model with *entities-original* and *opinions-original* for random and rule-based distractors in comparison to the dataset. All 5 categories were evaluated on a Likert scale with 5 levels (higher is better). Standard deviation is shown in brackets.

Human Evaluation For the best models listed in Table 4.9 we conducted a human evaluation to analyze overall dialogue quality, adherence to opinions and correctness of facts. We created another HIT on AMT with a minimum acceptance-rate of 95% similar to the dataset collection. To further protect against scam, we added two fake questions that each crowd-worker has to answer. We ask for the subject of the conversation (which is always *movies*) and for the name of the movie they talked about. Both questions can be evaluated automatically. We let 95 different crowd-worker generate and evaluate 240 conversations with our best models. In addition, we evaluated 120 dialogues from the original dataset. We did not mention that they talk to a bot, but simply ask them to chat about the given movie instead. Afterwards, they were tasked to rate the following five statements on a Likert scale with five levels from *strongly disagree* (1) to *strongly agree* (5):

1. Naturalness: The conversation felt natural.
2. Attentiveness: It felt like your chat partner was attentive to the conversation.
3. Consistency: The conversation was overall consistent.
4. Personality: The conversation fits well to the intended character described above.
5. Knowledge: Your partner replied correct to the asked questions.

We list results in Table 4.11. Statements for the dataset are all between *agree* (4) and *strongly agree* (5) which again proofs the quality of our dataset. The knowledgeability of our dataset is lower compared to the adherence to the personality. After investigating the lower rated dialogues, we found that sometimes participants ask questions that the dialogue partner could not answer. However, this did not happen often and 4.14 is still above *agree*. Most facts were used correctly in any statements and answers.

Both models perform well compared to the dataset baseline. The *rule-based distractors* model performs better in all categories. It appears that the classification logits help to choose a better contribution, as the better model has higher accuracy on next utterance classification. Values of 4.01 and 4.04 on personality and knowledge, respectively, show that models

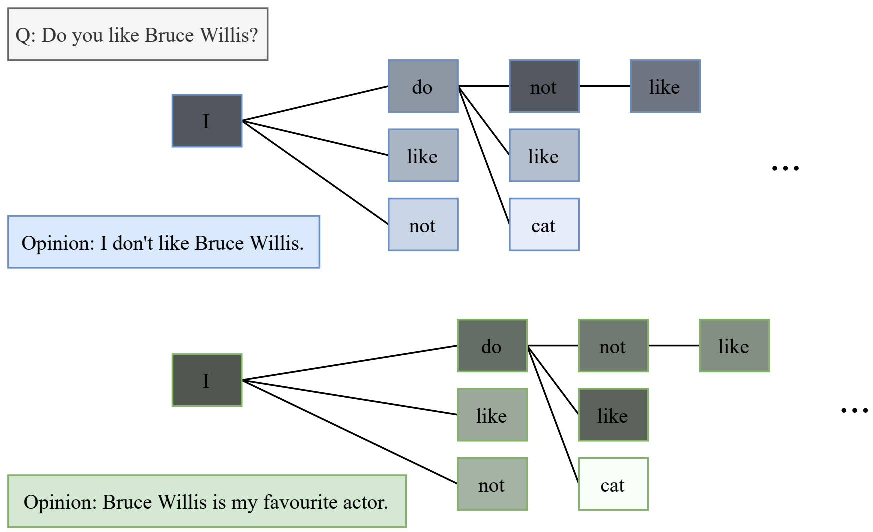

Figure 4.15.: Qualitative representation of token probabilities for two (partial) beams at inference on the KOMODIS dataset on the question: "Do you like Bruce Willis?". Lighter colored tokens represent lower probabilities. The token *cat* was added to show probability near zero for comparison.

trained on KOMODIS are able to learn consistency towards facts and opinions. When we hold conversations with our models, we do noticed a difference in quality between opinions and facts. Occasionally, the models answer with opposite opinions on named entities and models tend to slightly prefer including the *not* token into their contributions (e.g., "I do not like him" or "This is not my favorite genre!"). *Not* is a frequent token and therefore generally overestimated by our models. Looking at possible beams, this token always has high probability, regardless of the provided opinion. A perfect model, however, should have very low probability on the *not* token while expressing a positive opinion. This phenomenon is visualized in Figure 4.15. This also shows the problem of token-by-token decoding: At the step of expressing the opinion in that responses, the target entity is not yet known. A possible and valid response could have been: "I do not like Samuel L. Jackson, but Bruce Willis is great!". Without knowing what the model want to communicate at all, probability tokens cannot be estimated properly.

Models fine-tuned on KOMODIS cannot handle other topics than movies, as shown in Figure 4.16. We expected that, as the dataset is closed-domain and models cannot learn other domains without further effort. We will elaborate this in Chapter 7, where we investigate the influence of transfer learning techniques on closed-domain datasets. The dialogue example in Figure 4.17 shows a grammatical error probably caused by beam-search, but is something that seems to occur rarely. In general, for state-of-the-art neural conversation models, flu-

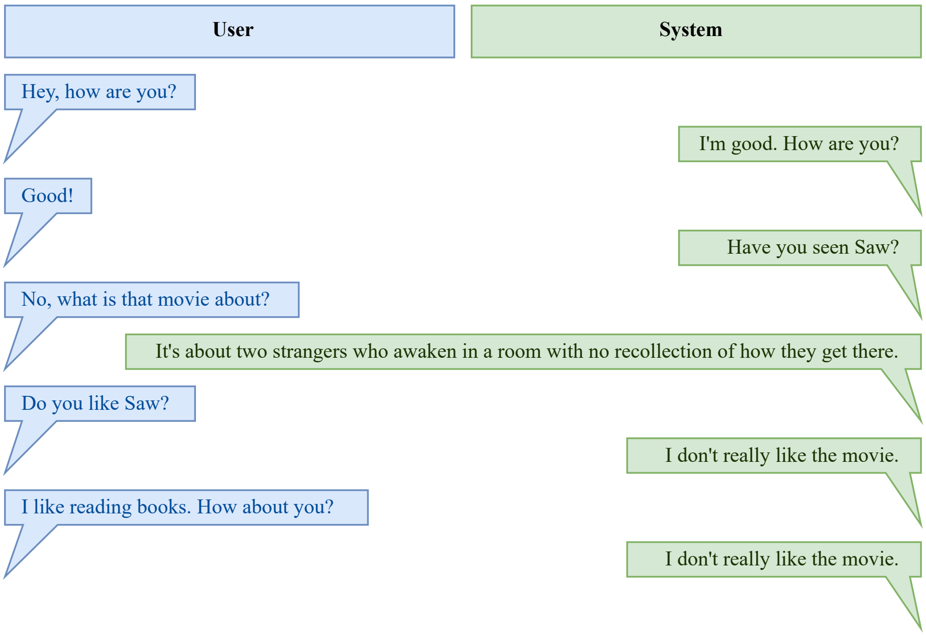

Figure 4.16.: A dialogue between a human and our model with random distractors from the human evaluation. The conversation about the movie Saw seems natural, but the model is not able to handle different topics (see last reply).

ency is not an issue at all. This could have been avoided by using some randomness in the decoding algorithm. However, as we will show in the next Chapter 5, randomness can cause issues with consistency.

4.4.6. Conclusion

In this section, we described our dataset in greater detail, validated the quality and showed experiments with different training strategies. The dataset is medium-sized, contains closed-domain dialogues augmented with correctly rendered speaker profiles and appropriate for fine-tuning language models to build neural conversation systems. It is easier to learn factual consistency than consistency towards opinions. Encoding the speaker profiles similar to the dialogue history is effective and works well. It is not necessary to replace named entities with delexicalisation tokens, as GPT and GPT-2 models are strong pre-trained language models that can handle entity generalization. When using KOMODIS it is important to only evaluate on dialogues about movies, as this dataset is not designed to train an open-domain conversation system. The additional classification layer is helpful at inference, but rules to make this training process more effective need to be determined carefully. Some randomness

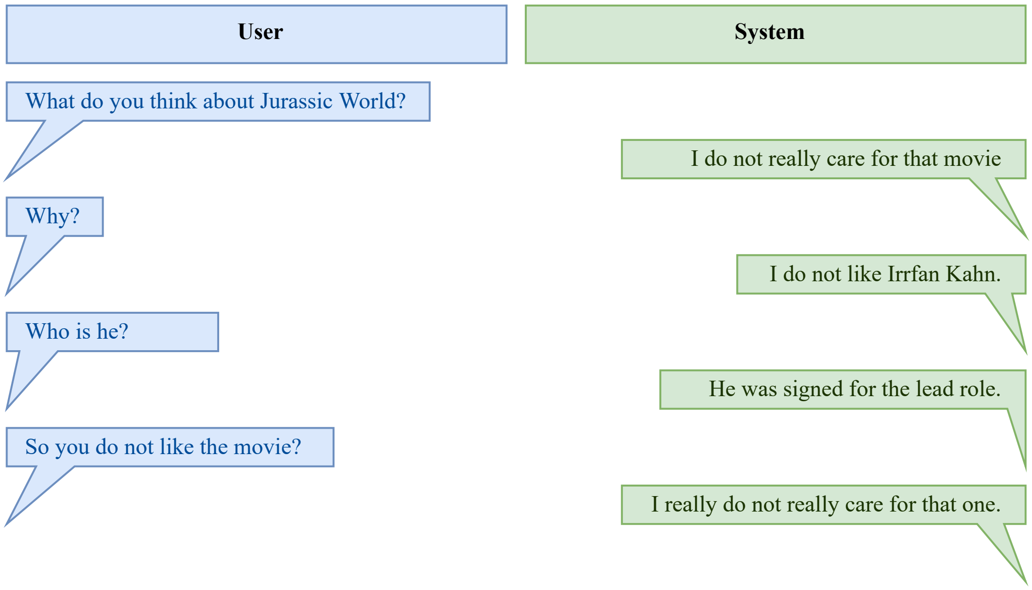

Figure 4.17.: A dialogue between a human and our model with random distractors from the human evaluation. The system adheres to a consistent opinion about the movie Jurassic World. The grammar in the last reply is wrong because of the loop over the word *really*.

should be included all the time. Otherwise, the classification goal might shift towards the rules that only used distractors from the same movie, which hurts the classification quality at inference, as observed in our experiments.

Conversations with models trained on KOMODIS look natural, and the facts are correct most of the time. However, at this point, we want to emphasize a few shortcomings: Learning to reason opinions through facts is not an easy task for a neural conversation model. When interacting with our trained models they do give explanations on why they lean towards a specific opinion, but this often seems random, for example: *"He is a great actor. He played in a local soccer team when he was younger."*. While this could make sense if the character represented by the model likes football, but it often just feels arbitrarily. The fact that these models lack long-term coherence amplifies that feeling of not really understanding the facts. However, we think this aligns well with previous results (as shown in chapters 2.3 and 3), where all these neural conversation models miss deeper complexity, when comparing them to actual human conversation partners.

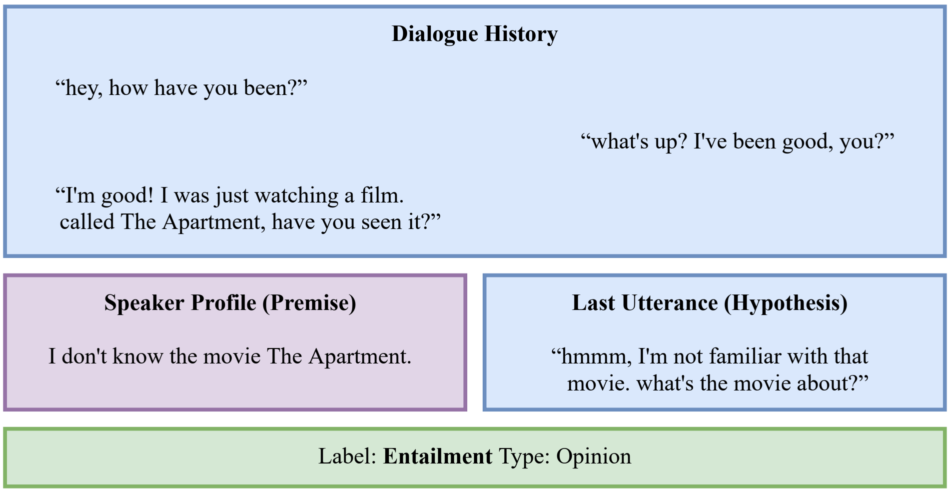

Figure 4.18.: Sample from the entailment statements extension data. Each contains a premise based on the speaker profile, a hypothesis as a next utterance and a dialogue history.

4.5. Entailment Statements Extension

Our KOMODIS dataset contains a lot of opinionated utterances (e.g., *“Do you like Pulp Fiction?”*, *“No I don’t!”*) and related opinions within the speaker profiles (e.g., *“I don’t like Pulp Fiction.”*) as well as fact-based utterances. We validated the adherence to these speaker profiles in Subsection 4.4.3. Estimating the alignment between an opinionated utterance (or a factual utterance) in a dialogue and a general opinion description (or fact) from the speaker profile can be seen as a natural language inference problem. In this setup, the opinion would be the premise and the utterance, given a dialogue history, would be the hypothesis.

Based on the data available through KOMODIS we create a new dataset based on the collected dialogues and speaker profiles. This new dataset is supposed to train a natural language inference model, that decides whether an utterance is consistent towards a fact or an opinion. Without such a classification model, automated evaluation of *consistency* is not possible. We will also make use of this dataset in Chapter 6 with reinforcement learning. Additionally, the labels we create in this process were used for the *rule-based distractors* in the previous experiments.

We first discuss the sentiment label annotation that we used to augment the dataset in Subsection 4.5.1. We conducted human annotators to create additional samples for the new dataset that we present in Subsection 4.5.2. Dataset statistics are presented in Subsection 4.5.4. We then show and discuss experiments and results in Subsections 4.5.5 and 4.5.6.

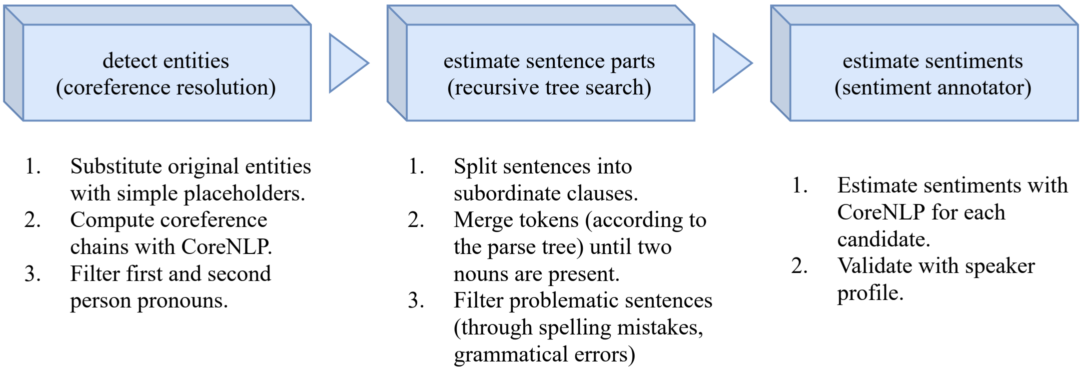

Figure 4.19.: Overview of our algorithm for automated sentiment labels on dialogues from KOMODIS. Utilized by the tool CoreNLP from Manning et al. (2014)

4.5.1. Automated Sentiment Labels

Creating utterance-level labels for expressing opinions requires multiple processing steps: Resolving indirect references (e.g., “I like it!” may need to be converted to “I like Pulp Fiction!”) to identify the target entity of all opinionated statements. Splitting sentences into all statements because we need to identify the smallest unit around each entity that can carry a sentiment. And lastly, a sentiment annotation for each statement. The whole process is summarized in Figure 4.19. In the following paragraphs we explain each part in greater detail.

Detect Entities To detect an opinion about a named entity, we first had to resolve indirect references. We use the co-reference resolution annotator from CoreNLP (Manning et al., 2014) for this task. However, for our specific case we noticed some problems with the CoreNLP heuristics, presumably because our data is different from its training data. After extensive manual investigation, we determined three problems: The annotator from CoreNLP has low accuracy on unusual names, lowercased names, long movie titles and (which was expected) entities with spelling mistakes. Indirect references to the movie by calling it *movie* or *film* were not resolved properly. And lastly, first and second-person pronouns are often grouped with entities together mistakenly.
We solved these issues by pre-processing named entities and a few specific rules. We replaced all direct mentions of entities (entities are labeled in our dataset) by placeholder tokens that work well with the algorithm:

```
MOVIE_RPL  = "Pulp Fiction"
PERSON_RPL = {0: "Steve",
```

```
  1: "Peter",
  2: "Phil",
  3: "Joey"}
COUNTRY_RPL = "United States"
GENRE_RPL   = {0: "comedy",
  1: "romance"}
```

Since all dialogues are about one specific movie, phrases like "*This movie is great!*" almost always refer to the given movie. We therefore resolved the following references manually:

```
MOVIE_REFERENCES = ["that movie", "this movie", "that film", "this
 film"]
```

To tackle wrong references caused by first and second pronouns, we simply filtered out all of them. This works because in a conversation these pronouns almost always refer to the dialogue partners and therefore never reference any of the movie entities. Given the following dialogue between person P_A and person P_B:

> P_A: "Do you know **Pulp Fiction**?"
> P_B: "Yes, it is a great movie!"
> P_A: "**Samuel L. Jackson** had a good performance in it."
> P_B: " That's for sure! He is my favorite actor."

the result of our specific co-reference resolution would be:

> P_A: "Do you know **Pulp Fiction?**"
> P_B: "Yes, **Pulp Fiction** is a great movie!"
> P_A: "**Steve** had a good performance in **Pulp Fiction**."
> P_B: " That's for sure! **Steve** is my favorite actor."

Estimate Sentence Parts To detect the named entity related sentiments, the smallest unit around an entity that can carry a sentiment needs to be identified. This is necessary because the target sentiment for the sentiment annotation is the full phrase. We, however, are interested in the sentiment towards a named entity. Figure 4.20 visualizes the issue. In that example, Pulp Fiction and Bruce Willis are classified with a negative sentiment. Only by separating the sentence, the use of the sentiment annotation as a classification towards each entity works as intended. To achieve this, we first split utterances into sentences and create full constituency parse trees with the help of CoreNLP. Then, the algorithm goes over all first-level Parts of Speech (POS) tags and runs recursive. The following pseudocode describes the core of the algorithm:

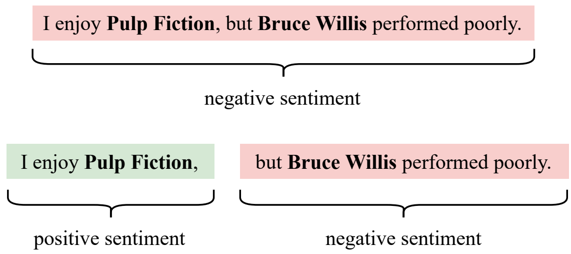

Figure 4.20.: Automatically estimated sentiments with models from CoreNLP. Named entities are marked in bold. The full sentence is classified with a negative sentiment, which causes an error for Pulp Fiction. If the sentence is divided, each clause (and therefore entity) is classified with the correct sentiment.

```
def analyse_tree(tree,     # A sentence tree object.
depth):   # An integer. Subtree depth.

# check if tree doesn't have subordinate clauses
if len(get_num_clauses(tree)) == 0:
# if tree has at least one noun or is a clause by itself
# stop search and return candidate.
nns = get_num_nouns(tree)
if is_S_or_SBAR(tree) or len(nns) > 0:
return [create_candidate(tree, depth)]
else:
return []

# If tree has subordinate clauses, analyse all subtrees.
subtrees = tree.child
subtrees_pr = []
for subtree in subtrees:
# recursive call for each subtree
subsubtrees = analyse_tree(subtree, depth+1)
if len(subsubtrees) > 0:
# if subsubtrees are available, add them to the
# processed subtrees.
subtrees_pr += subsubtrees
else:
# Subtree is not a candidate for a sentence part
subtrees_pr += create_non_candidate(subtree, depth)

# upwards merging candidates with non candidates until
# all words are covered.
return merge(subtrees_analysed)
```

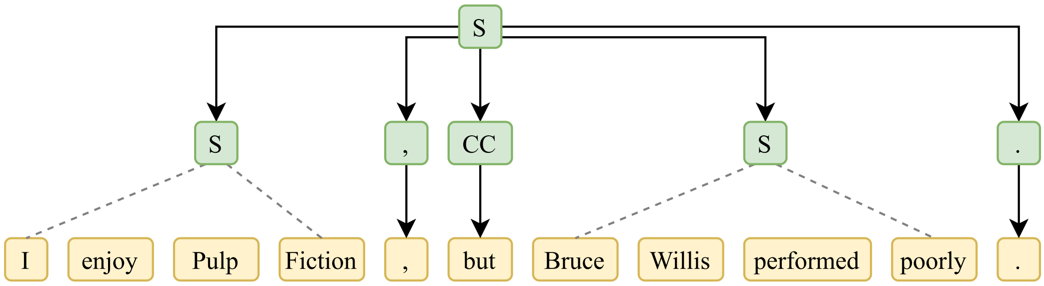

Figure 4.21.: Part of a constituency parse tree estimated with CoreNLP. *S* means simple declarative clause and *CC* means coordinating conjunction.

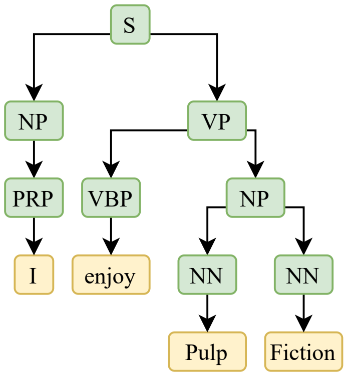

Figure 4.22.: Constituency parse tree estimated with CoreNLP. *S* means declarative clause and *NP* means noun phrase, *VP* means verb phrase, *PRP* means personal pronoun, *VBP* means verb non-3rd person singular present and *NN* means noun.

Given the tree in Figure 4.21 the algorithm would first analyze the left simple declarative clause ("I enjoy Pulp Fiction"), then a punctuation mark (","), then another simple declarative clause ("but Bruce Willis performed poorly") and finally a second punctuation mark ("."). If any of these have subordinate clauses within their trees, the *analyse_tree*-function calls itself for these trees. Any subtree that contains at least one noun (excluding personal pronouns as nouns) or is a declarative or subordinate clause by itself, is added as a separate candidate for a sentence part. Figure 4.22 is an example of a subtree that is a sentence part candidate. It has one noun (Pulp Fiction) and is a simple declarative clause by itself. Subtrees that do not fall under these categories and cannot be further split are added as *non candidates* (in this example the word "but") which are added to the candidates while running from back from bottom to top. The merging process for this example is shown in Figure 4.23. The word "but" is added to the right candidate, as they are in the same subtree. Two candidates cannot be merged at all, since they contain different nouns and the goal of the algorithm is to separate them. Multiple non-candidate parts can be merged to a candidate part, but the

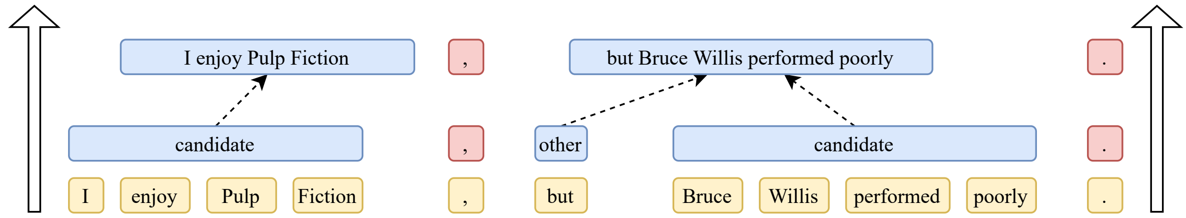

Figure 4.23.: Fractions of a sentence from our recursive tree search. Arrows show the merging process for the final sentence parts (blue, top line).

number of nouns (excluding personal pronouns) must not exceed 2. We choose 2 instead of 1 because sentences such as: "I do like him as an actor." or "I like the movie Pulp Fiction" do contain two different nouns. "I like Peter and Sara" is also one sentence part, as splitting them would remove the *I like* from *Sara*. This would not cause problems, as the sentiment for both is the same. A sentence like "I like Peter, but I do not like Sara" would be split by our algorithm before the merging process into two separate candidates, as we have two subordinate clauses in that case.
However, this algorithm does not cover all cases, especially if sentences have grammatical errors or spelling mistakes. We discuss the quality of this automated annotation at the end of this subsection.

Estimate Sentiments Given the sentence parts (ideally with one named entity per part) the next step is straight-forward. We use the sentiment annotator from CoreNLP which provides a discrete probability distribution over the five sentiment classes: *VeryNegative, Negative, Neutral, Positive, and VeryPositive*. If a sentence part does contain a named entity, we compare the result of the sentiment annotation with the speaker profile from the dataset. If they match, we add this label to the utterance of the dataset. The comparison step is necessary as we measured an accuracy for the automated sentiment labels of 80.1% when compared to the speaker profiles. 80.1% accuracy is unacceptable for labels, so we decided to only add annotations that do align with the speaker profile. This, however, leads to at least 19.9% missing labels. We used these annotations to create the successful *rule-based distractors* for the experiments from Section 4.5.5.

Due to the quality issues from our automated annotation, we decided to collect additional data with crowdsourcing for the new sentiment dataset. We describe the collection process in the next subsection.

4.5.2. Human Annotated Sentiment and Fact Labels

To complete our dataset and to make sure that we cover the maximum range of variety from the KOMODIS dialogues (the automated annotation might be biased) we create an additional data collection task with AMT. The instruction slides can be found in Appendix A. This time, a HIT can be done without interacting with another participant, so a slurk setup is not needed. We design a HIT so that a participant has to read a dialogue and a speaker profile. He then has to choose all facts and opinions that are used in the last utterance of the dialogue history. We show up to four previous utterances and the last reply (left box, marked in bold) to the participant. To make the task more intuitive, we decided to name both participants (Sam and Bob; Sam is always the person that contributes with the last utterance). On the right, we present facts and opinions and define them as “Sam's possible knowledge” and “Sam's possible opinions”. We don't use the expressions inference, hypothesis and premise, but instead ask them to choose what Sam must have been thinking or knowing to make the task easier to understand for the participants. In the second instruction slide we present specific cases with correct annotations to make the crowd-worker aware that details are important to do this task successfully.

In our first tests we experienced disagreement between participants, which appears to be common. Similar data collection setups like the Stanford Natural Language Inference Corpus Bowman et al. (2015) used redundant annotation (here, every sample was evaluated by 5 different participants) to find the most appropriate label. Additionally, we need scam protection as usual. We decided to use *trap facts and opinions* from which we know are definitely wrong, as well as simple pre-filtering of crowd-worker like in the KOMODIS collection process. We also collect labels from three different crowd-worker per sample. With this setup we only collect entailments. We plan to augment the data automatically with neutral samples and contradictions, which we describe in the next Subsection 4.5.3.

Data Preparation In order to create HITs for the crowd-worker, we have to preprocess the data: We only create samples from dialogues with above-average number of turns and tokens per utterance. The data quality varies between the dialogues and shorter dialogues tend to have less quality. Further, long dialogues provide richer utterances and on average higher diversity. Then, we collect all facts and opinions from the speaker profile and augment the opinions by opposite opinions so that each sample has a total of five opinions. Opposite opinions are created by using opposite sentence templates, e.g., “I like Pulp Fiction” is transformed into “I don't like Pulp Fiction”. If the number of opinions is less than three, we fill the five slots by randomly copying one of the opinions. This can be seen as another scam check because if two opinions are identical, either both must be selected or neither of them. One sample contains up to five utterances from one dialogue from the same speaker, always providing the same facts and opinions. We underline named entities in the dialogue history and in the facts and opinions (this is simple, as named entities are annotated in KOMODIS) to make the labeling process as clear as possible.

Post Processing We collect three annotations for each sample and spread the collection over two weeks. This is necessary to manually watch the quality of the annotations. We count the number of selected *trap facts and opinions* to block crowd worker that doesn't match our expectations. In addition, we only use data where at least two crowd-worker agree on and recollected samples that were annotated by crowd-worker that we blocked from the task afterwards.

4.5.3. Neutral and Contradiction Statement Augmentation

Combining our data from the automated annotation and AMT we have a total of 844 samples with the label *entailment*. The next step is to augment these samples with *neutral* and *contradiction* labels. We achieve this by swapping opinions and facts to create contradictions, and by swapping entities or adding new facts and opinions for neutral statements. To balance the dataset, we decided to create a similar number of contradictions but a significant larger amount of neutral statements as they occur more often since usually an utterance is only about one specific entity.

Opinion Swaps Under the assumption that a premise about an opinion regarding a named entity entails with the hypothesis (here utterance in the dialogue), a swap of that opinion in the premise must cause a contradiction with the hypothesis. We therefore create one opinion contradictions for every opinion entailment available. Possible swaps are *positive* $\rightarrow$ *negative*, *negative* $\rightarrow$ *positive*, *unknown* $\rightarrow$ *positive or negative* and *positive or negative* $\rightarrow$ *unknown*. We don't differentiate between the different strengths for positive and negative opinions, but sampled them randomly with the same probability distribution that we used to create the speaker profiles.
To create neutral statements, we have to swap the target entity of the opinion. However, we have to make sure that, when we use a different target, this target is not present in the utterance, as this could cause an entailment or a contradiction: Given the hypothesis: "I think

both Bruce Willis and Samuel L. Jackson are bad actors" and the entailing premise: "I really don't like Bruce Willis", a target swap to: "I really don't like Samuel L. Jackson" would still be an entailment. To make sure that the resulting samples are neutral, we either only take samples from which we know that this is not the case (samples from AMT that meet this requirement and without any disagreement between all three annotators) or by using target entities that are not used in the dialogue at all. We assume that the latter is easier to learn for a neural network and less important, since at inference facts and opinions are always about occurring entities.

Fact Swaps Similar to the idea of opinion swapping, changing a fact from an entailing premise hypothesis pair must result in a contradiction. However, this does not work for facts with a content range greater than one: A movie has multiple actors, therefore swapping to another actor would only make the sample neutral. That would actually be the case for most of our facts, but we decided to assume that there is only one director, one writer and one shot location for each movie. In addition to these, we create contradictions by swapping the movie plot, release year and age restriction to values from different movies.
To create neutral statements, we swapped the type of the fact. Similar to the opinion statements, we have to make sure that these facts are not present in the utterance. Again, we use the data from the AMT collection for this, but also add facts that aren't part of the whole speaker profile. From our dataset investigations we know that participants never spoke about other facts than the one from the speaker profiles. To complete the set of neutral samples, we also swapped the movie title for some of them to make sure that models trained on this data also learn to take the movie title into account.

Given the original 844 entailment samples, we were able to augment the dataset with 811 contradicting statements and 2,028 neutral samples.

4.5.4. Dataset Statistics

The final dataset consists of 3,683 statements, with 2,254 being about an opinion towards an entity. Statistics are summarized in Table 4.12. For each sample the full dialogue history and fact or opinion information is available. The inference of hypothesis and premise pairs is dependent on the dialogue history, which makes this task hard to solve. In addition to the task of natural language inference in general, the model also has to resolve coreferences as the target entities are often referenced indirectly in the last utterance. Since the KOMODIS

Entailment Statements extension statistics	
samples (total)	3,683
entailments	844
contradictions	811
neutral statements	2028
fact statements	1,429
opinion statements	2,254

Table 4.12.: Dataset statistics for our Entailment Statements extension.

dataset is domain-specific, a lot of the dialogue content is about opinionated entities and facts, which makes the task even harder. The model needs to differentiate between relevant and irrelevant entities.

1,262 samples come from the KOMODIS dataset directly. 349 of these samples are entailments and created with the help of our automated annotation. Every sample was carefully validated manually to ensure its correctness. The 913 additional samples were generated out of the entailments as explained in the subsection above. The remaining 2,421 samples come from additional crowdsourcing. The 495 entailments were not controlled manually, but we secured quality by our post-processing rules explained in Subsection 4.5.2. From these entailment statements we generated the additional 1,926 neutral statements and contradictions automatically with the rules explained in the previous subsection.

In the next subsection, we train basic neural networks to validate and analyze the new dataset.

4.5.5. Experiments

For our experiments we use the RoBERTa model (Liu et al., 2019), a BERT model fine tuned on the MultiNLI dataset (Williams et al., 2018). More details about this model and the dataset can be found in the Fundamentals Section 2. To utilize the pre-trained weights from RoBERTa as much as possible, it is necessary to define an encoding that is as similar as possible to the one used for RoBERTa. This encoding uses three special tokens: a *begin-of-sequence* token, an *end-of-sequence* token and a *separator* token. After the *begin-of-sequence* token, the premise is added to the sequence, followed by two *separator* tokens. Then, the hypothesis is added, followed by the *end-of-sequence* token. Figure 4.24 shows an example of this encoding. For our purposes we reuse the *separator* token, but have to add another two special tokens: a *speaker-turn* token that signalizes a speaker turn in the dialogue

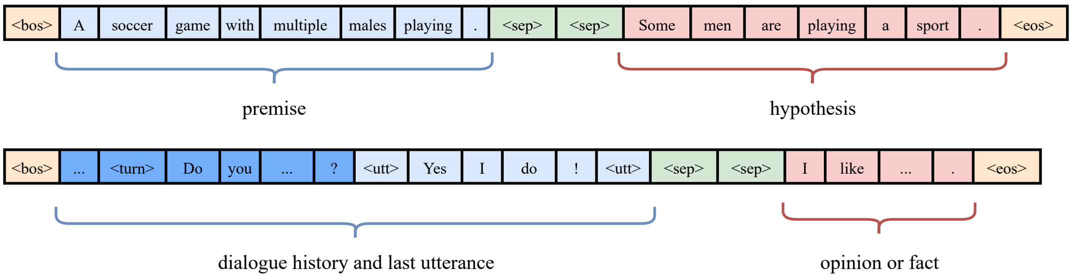

Figure 4.24.: Sequence encoding from the RoBERTa model (top, example taken from the MultiNLI dataset) and adapted encoding for our Entailment Statements dataset (bottom).

history and an *utterance* token that signalizes the last utterance that is supposed to be the target utterance for the classification task. Our version of the encoding is also visualized in Figure 4.24. We put the fact or opinion from the speaker profile after the separation tokens, similar to the hypothesis of the RoBERTa implementation. The *utterance* tokens around the last utterance are theoretically not necessary, as the model could learn the meaning of the last utterance by itself, but we decided to add these token to make that step of generalization less difficult for the model.

In the original sense of inference a hypothesis entails with a premise, if the hypothesis is true whenever the premise is true, but not necessarily vice versa; concluding that they are not exchangeable: If the hypothesis is true, the premise does not need to be true. In our case, however, it is similar in both directions, and thus we could see both as the premise (i.e., encoding it on the left side of the *separator* token). We encode the dialogue history and last utterance as the premise for two reasons: It minimizes the distance between the important last utterance tokens and the fact or opinion, and the dialogue history contains a lot of information that cannot be concluded from the hypothesis, which is a similar non-symmetric characteristic like the original relation between premise and hypothesis.

Data Split Since our Entailment Statements extension is small, we decided to use five-fold cross-validation as a training strategy instead of a standard train-validation data split. We put 15% of the dataset samples away as a test set and used the remaining 85% of the data for training and cross-validation. For each fold, 20% of the training data is used for validation. After five folds are done, we averaged the number of optimal epochs (the lowest validation loss) to estimate the number of epochs for the final training step. More details about n-fold cross-validation can be found in the Fundamentals Section 2. To create a balanced test set we split entailments, neutral statements and contradictions separately, as the low number of

batch size	**learning rate**	**history length**	**epochs**	**accuracy**
8	$5 \cdot 10^{-6}$	2	6	82.23
16	$5 \cdot 10^{-6}$	2	4	83.04
32	$5 \cdot 10^{-6}$	2	4	**83.77**
64	$5 \cdot 10^{-6}$	2	4	77.21
16	$5 \cdot 10^{-6}$	2	4	83.04
16	$1 \cdot 10^{-5}$	2	3	84.53
16	$\mathbf{2 \cdot 10^{-5}}$	2	4	**84.70**
16	$4 \cdot 10^{-5}$	2	4	84.57
16	$8 \cdot 10^{-5}$	2	6	79.54
16	$5 \cdot 10^{-6}$	3	5	82.87
16	$5 \cdot 10^{-6}$	2	4	83.04
16	$5 \cdot 10^{-6}$	**1**	6	**83.29**
16	$5 \cdot 10^{-6}$	0	3	80.57

Table 4.13.: Experiments with 5-fold cross validation on the Entailment Statements dataset. Epochs and accuracies are averaged over each fold. The goal of this series is to analyze the influence of the parameters batch size, number of utterances in the dialogue history and learning rate.

samples might cause an imbalance resulting in non-representative results on the test set.

Hyperparameter Analysis We use the same model and train parameter from the RoBERTa setup by default. To analyze our dataset, we vary batch size, learning rate and number of utterances in the dialogue history. Especially the latter gives interesting insights in what the model learns specifically. Our assumption is that at least one additional utterance is required to get good results. Since references can't be resolved if these utterances are not available, it is impossible for the model to differentiate between neutral and non-neutral statements. However, having too many utterances in the history should also lower the result, as more distracting statements are present in the training sample. Batch size and learning are very important hyperparameter, but since the model is already pre-trained, optimal values should be in close range to the pre-training ones.

We list all results in Table 4.13. We only vary one hyperparameter at a time (instead of a 3-dimensional search) since this requires much less resources, and we are especially interested in the influence of each parameter individually.

batch size	learning rate	history	accuracy			
			overall	entail	neutral	contra
32	$2 \cdot 10^{-5}$	1	84.59	87.33	82.37	84.26
32	$4 \cdot 10^{-5}$	1	85.95	**93.66**	80.32	84.26
32	$2 \cdot 10^{-5}$	2	**89.57**	93.21	**85.56**	**90.35**
32	$4 \cdot 10^{-5}$	2	86.49	92.48	83.17	86.73

Table 4.14.: Results on test data for experiments on our Entailment Statements dataset. In addition to the overall accuracies we measured each class on the test set individually.

Per-class Accuracy The Entailment Statements dataset has a class imbalance, because we increased the number of neutral statements to better fit reality. This can cause overfitting on the overrepresented class, as the that bias is higher and therefore choosing this class has a higher base success rate. For our final experiments we analyze the per-class accuracy on the test set to get insights in the training process. All results are listed in Table 4.14.

4.5.6. Results

In the following subsection, we present and discuss the results from our experiments. Results are listed in Tables 4.13 and 4.14.

Batch Size and Learning Rate The common batch sizes of 16 and 32 work well for our dataset. In our experiments a batch size of 32 produced a 0.7% higher accuracy, but we rate this as in and therefore can't conclude that one should always prefer 32 over 16. However, lower or higher batch sizes are not recommended, as the accuracy drops significantly in both directions. The results in Table 4.13 are not listed in chronological order, and we defined batch size 16 as the default value before this experiment series. However, for the final experiments, we use a batch size of 32.

The right learning rate is important to maximize training efficiency. We found that the optimal learning rates for our dataset and model setup are between $1 \cdot 10^{-5}$ and $4 \cdot 10^{-5}$. Increasing the learning rate further reduces the accuracy significantly, while lowering the learning rate has less negative impact on the model performance. The values are common for fine-tuning Transformer-based models. Again, our dataset is small and therefore we cannot estimate one specific learning rate that performs best.

History Length While batch size and learning rate are common parameters for any fine-tuning task, the number of utterances that are used in addition to the target utterance is a specific and important parameter to our setting. Not using any dialogue history results in an accuracy of about 80%. This indicates that there is a correlation between content of the last utterance and content of the dialogue history. The improvement of about 3% accuracy when adding the previous utterance is therefore also lower than expected. The accuracy increases because the model as access to important new information of the dialogue history. At the same time, we assume that new phrases from that utterance increases the difficulty of the task, since they act as distractors (i.e., a phrase could have an opposite opinion about another entity, making the classification harder). Adding more utterances to the history does not increase the accuracy. Both effects seem to nullify each other: More information causes more distraction, but also increases the available amount of necessary information.

Final Model Performance We chose four different settings for our final trainings and list all results in Table 4.14. Strictly following the train, validation and test split we have to report an overall accuracy of 84.59% (model from the first row) as this is the model with the best parameters according to our experimental series. We see, however, that with other good parametrizations we can increase the accuracy on test set to up to 89.57%. However, it is very likely that this model is over-fitted on the test data.
It appears that it is easier for the model to detect entailments (accuracy between 87% and rounded up 94%) than contradictions (accuracy between 84% and 90%). And even though there is an imbalance towards more neutral utterances, classifying them is the hardest part of this dataset (accuracy between 80% and 85.5%). In general, this shows that these models only slightly over-fit on neutral statements, which is important for using the final models in future experiments. We do found trainings where the model weights fall into a local optimum, only predicting the neutral class. This happens more often when using bad a parametrization and never happened on any of the four parametrizations from this series.
The performance of our models is with 85% to 89% accuracy not far behind the RoBERTa model on the MultiNLI dataset with 90.8%. The largest error source across all experiments is the misclassification of neutral statements, even in the sets with an increased amount of neutral samples. The addition of a dialogue history makes the task hard, as the model has to learn to differentiate between the dialogue partners. This was never learned during pre-training. Further, we assume that increasing the dataset size improves accuracy, since our dataset is very limited. In Chapter 6 we will apply our best models to a reinforcement learning setup. These experiments will show, if the model performance is high enough for automated evaluation and appropriate feedback loops.

5. Efficient Context Representation with Transformer Architectures

This chapter is based on our publication: Space Efficient Context Encoding for Non-Task-Oriented Dialogue Generation with Graph Attention Transformer (Galetzka et al., 2021). I was responsible for creating the graph encoding, adapting the Transformer model, conducting the experiments and writing the according chapters. Jewgeni Rose was mainly responsible for creating and augmenting the knowledge graphs and writing the corresponding chapters. The idea of our approach is equally creditable to me and Jewgeni Rose. Some content from it appears verbatim here and will not be specifically credited in this thesis.

Utilizing dialogue generation with additional knowledge or personality descriptions requires that the underlying model can distinguish between different input data types. In the previous chapter we used additional separation tokens to be able to augment the input sequences with additional context to reuse pre-trained Transformer-based decoder architectures. We found that this works well for our KOMODIS dataset, but this approach has some drawbacks. The sequence length of a dialogue increases linearly with the number of utterances and, in practice, old utterances become more and more irrelevant to the current dialogue state. This keeps the required space for an input sequence quite low. Knowledge, however, requires much more space compared to dialogues, especially if the neural conversation model should be able to choose the right knowledge out of a knowledge base.

In this chapter, we elaborate the problem of space complexity from knowledge graphs and introduce a concise graph encoding by expressing the graph connections through restrictions on the attention weights. We first discuss limitations of context-augmentation for neural conversation models in Section 5.1. Then, we explore the characteristics of knowledge graphs and explain how we create and augment knowledge graphs for our experiments in Section 5.2. Section 5.3 gives an overview of our new model architecture and graph encoding. Experiments are documented in Section 5.4, and we discuss automated evaluation in Section 5.5. In order to evaluate our approach according to our needs, we conduct a human evaluation, which we present in Section 5.6.

5.1. Limitations of Context-Augmentation

The idea and purpose of context-augmentation for neural conversation models is straightforward: By providing specific knowledge and personality descriptions with a dialogue representing that context correctly, a model can learn to utilize such context in order to generate appropriate dialogues. But this creates some challenges to solve.

Pre-Training Mismatch State-of-the-art neural conversation models have an underlying language model with the purpose of modelling a continuation of a dialogue history. They are pre-trained on large plain text corpora, and their strength lies in generating fluent text conditioned on the input. Dialogue datasets with context-augmented dialogues (like KOMODIS) provide training data with the purpose of teaching the task of dialogue generation conditioned on a given context. Pre-trained language models don't have an architecture that allows to put such context into the model, and they did not learn to generate content explicitly based on context in the form of specific information. We try to minimize this mismatch by designing a fine-tuning task that is as similar as possible to the pre-training task while still fulfilling the requirements to efficiently learn dialogue generation conditioned on additional context.

Context Size At inference, the neural conversation model has to have access to any knowledge information and personality description available because we do not know what the user of this system is going to ask or say. Using current state-of-the-art architectures (Transformer-based or RNN-based models) are incapable of handling large external information at all. To manage this amount of data, some kind of knowledge retrieval algorithms are needed that preselect potential relevant context. We will discuss these algorithms in the next subsection. Here, the challenge is to get all relevant information because if some important fact or description is not chosen by these algorithms, the neural conversation model cannot condition on this information. Increasing model capacity for context reduces that risk. However, Transformer-based language models have strict input length restrictions, which we discuss in Subsection 5.1.2.

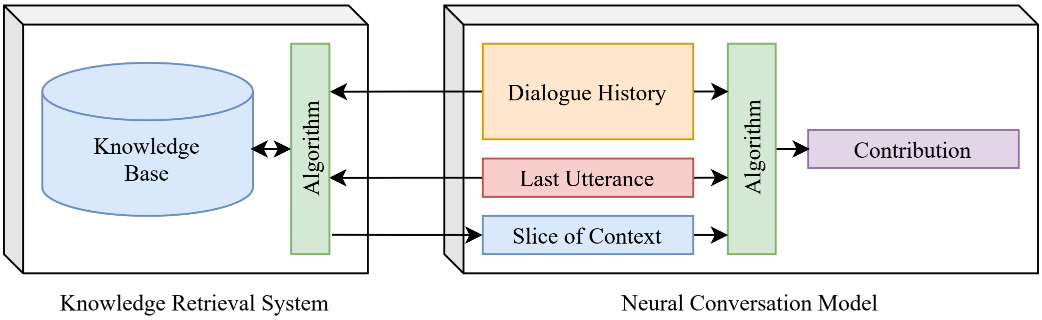

Figure 5.1.: A simplified overview of a knowledge retrieval system interacting with a neural conversation model. The model can only process a subset of context and therefore a separate system is needed to choose appropriate information out of a large knowledge base. In this setup, the neural conversation model algorithm has no influence on the slice of context at all.

5.1.1. Knowledge-Retrieval Systems

Large knowledge sources can be added to the neural conversation model by using a knowledge-retrieval system. The very basic idea of such systems is to automatically select an appropriate slice of knowledge from the source given a specific piece of information. In our case, this could be the dialogue history and last user utterance, for which an algorithm chooses a set of factual or personality information that might help the conversation model to generate an appropriate reply. The interaction between them is visualized in Figure 5.1. Many algorithms and architecture ideas exist, and Section 3.4.3 gives a technical overview of these systems.

Knowledge-retrieval systems play an important role in neural conversation models and since the techniques and tools that are needed to build them differ significantly from what is needed to create a neural conversation model, it does make sense to separate them. However, this separation obviously causes challenges: It is crucial that the retrieval system works well, since the conversation model only sees the slice of context provided by it. This creates a training inference discrepancy. At training, the neural conversation model has access to the optimal context provided by the data set, while at inference the model has to deal with the performance level of the retrieval system. In addition, one can ask: Are knowledge-retrieval systems capable of anticipating what the neural conversation model wants to say? If a user asks a question, the retrieval system needs to find the answer in the database, but what happens if the neural conversation model wants to proactively change the topic? While it might not be the neural conversation model's task to choose the right knowledge out of a large database, we do think that the conversation model has an appropriate amount of context choices in order to create a good contribution.

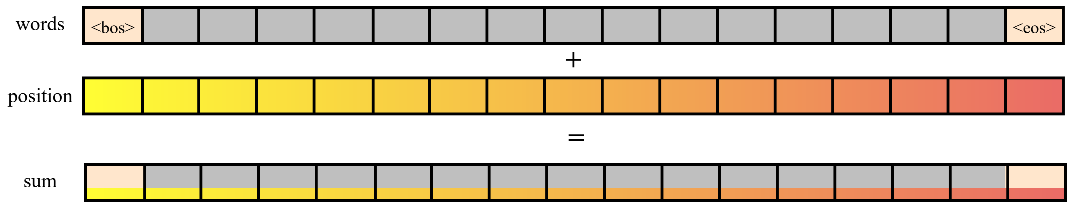

Figure 5.2.: Positional encoding idea for Transformer models. Since the input sequence is position invariant, a learned set of positional encodings is added token-wise to the word tokens. This allows the Transformer model to do position dependent computations.

The focus of this chapter is not about discussing the best architecture for incorporating large data sources, but to empower neural conversation models (specifically Transformer-based models) to process an appropriate amount of data in a convenient way.

5.1.2. Context Encoding with Transformers

Transformers are sequence-to-sequence architectures and therefore transform an input sequence (in our case the dialogue history) into an output sequence (the contribution) by predicting the next token in the sequence until an end-of-sequence token is predicted. For dialogue generation, it is common to pre-train the Transformer models on large text corpora (not necessarily dialogues) like the GPT-2 model. An overview of how Transformers work can be found in the Fundamentals Chapter 2.

Since Transformers are based on attention mechanisms that sum up values from the previous layer, their processing is order invariant. That means a pure Transformer model cannot process a sequence in a specific order. The order of words in a sentence (here tokens), however, is essential. That's why Transformer-based language models come with a set of pre-trained position encoding tokens that are added to the real tokens. At pre-training, the Transformer has learned to recognize the order of tokens by looking at these positional encodings. See Figure 5.2 for a simple example. This creates an upper bound of tokens a Transformer model can process, since new position encoding tokens can't be added during fine-tuning. Typical maximum sequence length are between 128 and 1,024 tokens. Depending on how much dialogue history should be kept, the amount of space for additional context is very limited. We will provide more detailed insights in future sections.

A knowledge base does not necessarily consist of plain text information. It is also common to provide knowledge in the form of lists or graphs with discrete attributes. Transformer input, however, is limited to a one dimensional sequence of word tokens, even if different types of Transformer architectures are used to enhance a neural conversation system. Therefore, any structured form of knowledge has to be transformed into a sequence of tokens. One example from our previous experiments is the encoding of speaker profile opinions. Instead of using abstract representations of the degree of likeliness, we use fully written sentences (that we had to generate) instead, which improve the performance of these models.

By simple concatenation of dialogue history and additional context, the Transformer model cannot distinguish between them. To solve that, the sequences need to be labelled by additional tokens, or a different Transformer architecture is needed, where context input is fed in separately. The latter causes a pre-training problem as the new architecture weights need to be initialized randomly, which might make the fine-tuning process significantly more difficult.

5.2. Knowledge Graph Representations

To conduct the experiments with graph encodings, we need appropriate datasets. Therefore, we process the KOMODIS dataset again in order to get knowledge graph-based dialogues. Since KOMODIS is a closed-domain dataset, and we also want to investigate the challenges that come with open-domain knowledge graphs, we use the OpenDialKG dataset (Moon et al., 2019) in addition. More details about this dataset can be found in Subsection 2.3.2. Before we explain the graph creation process in Section 5.2.2 we discuss the advantages of them over plain text sources in Section 5.2.1. To evaluate the influence of graph sizes, we define and create different sub-graphs for each dialogue in Section 5.2.3. In the last Section 5.2.4 we discuss the length requirements, i.e., space complexity, of our graphs and show that information is significantly compressed especially for larger knowledge sources.

5.2.1. Advantages over Unstructured Plain Text

Any kind of knowledge can be expressed in a structured way in the form of knowledge triples: subject, relation and object. These triples can be combined to a graph, where subjects are the graph nodes and objects are attributes, with the relation encoded in the edges of the graph. A small plain text source of information from a KOMODIS dialogue and it's

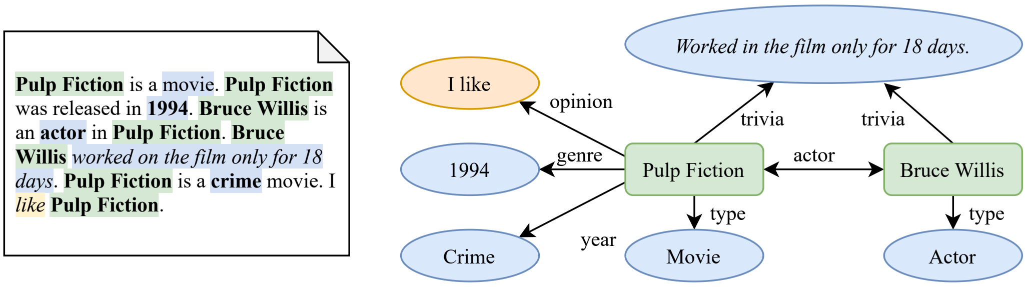

Figure 5.3.: The same knowledge encoded as plain text (left) and as a knowledge graph (right). Main entities are encoded as graph nodes (green) with their fact-based attributes (blue) and opinions (orange).

corresponding graph is shown in Figure 5.3. Here each sentence can be expressed by a triple: "Pulp Fiction is a movie" equals ⟨'*Pulp Fiction*', '*is of type*', '*movie*'⟩. This 'one sentence equals one triple' mapping might not always be the case for other data sources, but since KOMODIS was created by paraphrasing information from a database, the plain text is very similar to the corresponding triples. The main difference between a set of triples and their graph is, that in a graph each subject is only used once, see the following example:

1. ⟨'*Pulp Fiction*', '*is of type*', '*movie*'⟩
2. ⟨'*Pulp Fiction*', '*has genre*', '*crime*'⟩
3. ⟨'*Pulp Fiction*', '*has release year*', '*1994*'⟩

The node 'Pulp Fiction' is used three times in the list of triples, but only once in the graph in Figure 5.3. If nodes share attributes like the trivia information ("Bruce Willis worked in the film Pulp Fiction only for 18 days."), it also needs to be added once to the graph. The related triples would be:

1. ⟨'*Pulp Fiction*', '*has trivia*', '*'Bruce Willis worked ... only for 18 days."*'⟩
2. ⟨'*Bruce Willis*', '*has trivia*', '*'Bruce Willis worked ... only for 18 days."*'⟩

The edges between nodes and attributes can also be compressed when encoding a graph. This is not possible in a visual representation, since every edge needs a property, but when encoding the graph for machine processing one could replace edges with pointer towards one unique instance of that edge property, to further compress the information. These processing steps from triples to a graph reduce the required space per information and therefore sequence length. We estimate exact numbers in Subsection 5.2.4.

Another advantage of graphs is that they have information encoded in the structure itself. In plain text, information is unstructured and can be expressed in many ways and combinations: "Pulp Fiction is a crime movie" means the same as: "Pulp Fiction is of genre crime". Here, information is also given implicitly. The first sentence, for example, requires the machine to understand the concept of genre, since it is not explicitly stated in that first sentence. When processing plain text, the machine also needs to resolve references: "That is a great movie." Plain text can also contain distracting information: "Pulp Fiction was released in 1994 after Quentin Tarantino finished the writing in 1993." In a graph, the year '1993' would be an attribute of 'Quentin Tarantino' and cannot distract from the release year property of Pulp Fiction. Triples and especially knowledge graphs provide information in a reliable and structured way. If a machine knows how to estimate the release year of a specific movie, this knowledge can be generalized easily to other movies, since the structure is similar between all movies. In addition, knowledge retrieval systems can find needed information by using graph search algorithms (briefly explained in Section 5.1.1). Another advantage of graphs is that related content can be determined by using its structure. Other crime movies, for example, are close to 'Pulp Fiction' since they all point towards the attribute 'crime'. All these properties of graph encodings should give a machine an advantage over plain text for knowledge retrieval.

5.2.2. Knowledge Graph Creation

To create the graphs, we use the augmented context per dialogue as the base information for both datasets. Since both datasets have data structure and properties, we look at each dataset individually, starting with KOMODIS. In order to make use of our graph attention algorithm that we will introduce later, we have to create knowledge graphs that don't have descriptive edges. The relation between two nodes must be derivable from their nodes. We will explain this further in this subsection.

KOMODIS A speaker profile consists of a set of facts about a specific movie and a set of opinions towards the movie entities. We convert all facts into triples where the subject is either the movie or a person (actor, writer or director). Since we have a fixed and relatively small set of fact types (except for the trivia) we can create the full set of relation classes beforehand:

Speaker Profile

Facts

Pulp Fiction has actor: **Bruce Willis**.
Pulp Fiction release year: **1994**.
Pulp Fiction genre: **crime**.

Opinions

I like **Pulp Fiction**.
I like **crime** movies.

Figure 5.4.: A (made-up) speaker profile from the KOMODIS dataset.

- ‘has-genre’
- ‘has-release-year’
- ‘has-budget’
- ‘has-shot-location’
- ‘has-actor’
- ‘has-writer’
- ‘has-director’
- ‘has-age-certificate’
- ‘has-plot’
- ‘has-trivia’

We handle all trivia information as attributes of the type ‘trivia’. Therefore, we only need one additional relation class ‘has-trivia’ to cover the whole trivia content. For the KOMODIS graphs each node only has one attribute (‘is-a’) that defines the node type. In our specific case the relation between two nodes is given by their node types: If a movie node is connected with an actor node, it always means that the actor plays in that movie. In a more generic case this must not hold since an actor could also *watch* the movie: (⟨‘*Bruce Willis*’, ‘*is watching*’, ‘*Pulp Fiction*’⟩, for example.

For the opinions we only use one additional attribute type ‘opinion’. In the speaker profiles the property ‘unknown’ (“I don’t know Pulp Fiction”) is encoded as a specific opinion value. Instead of using this value, we decided to remove nodes from the graph, if the profile says they are unknown to the speaker. Given the speaker profile from Figure 5.4 the triples would be as follows:

- ⟨‘*Pulp Fiction*’, ‘*is-a*’, ‘*movie*’⟩
- ⟨‘*Pulp Fiction*’, ‘*has-actor*’, ‘*Bruce Willis*’⟩
- ⟨‘*Pulp Fiction*’, ‘*has-release-year*’, ‘*1994*’⟩
- ⟨‘*Pulp Fiction*’, ‘*has-genre*’, ‘*crime*’⟩
- ⟨‘*Pulp Fiction*’, ‘*opinion*’, ‘*like*’⟩
- ⟨‘*Bruce Willis*’, ‘*is-a*’, ‘*actor*’⟩
- ⟨‘*1994*’, ‘*is-a*’, ‘*release year*’⟩

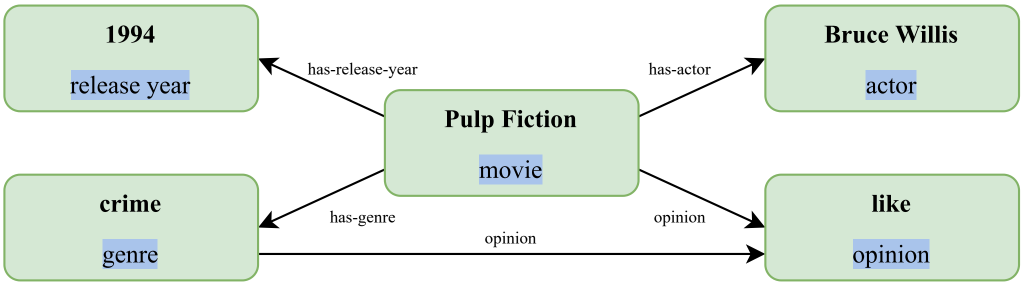

Figure 5.5.: A knowledge graph with 5 nodes (green). Each node has one attribute (blue). For this graph, relations between nodes can be derived from the node attribute.

- ⟨‘*crime*’, ‘*is-a*’, ‘*genre*’⟩
- ⟨‘*crime*’, ‘*opinion*’, ‘*like*’⟩
- ⟨‘*like*’, ‘*is-a*’, ‘*opinion*’⟩

Then, these triples are combined to the knowledge graph in Figure 5.3. As explained earlier, this graph (and all other KOMODIS graphs) has two special properties. First, each node only has one specific attribute that describes the node type and second, the relations between nodes can be derived by the node type. We will make use of this redundancy later in our experiments.

OpenDialKG This dataset already has knowledge triples as context. This time the relation between nodes is open-domain and therefore we cannot prepare a set of relations. Instead, we have to process them in a generic way. Given the following set of triples from one sample of the dataset:

- ⟨‘*The Secret Life of Bees*’, ‘*has_genre*’, ‘*Teen drama*’⟩
- ⟨‘*Teen drama*’, ‘ *has_genre*’, ‘*A Walk to Remember*’⟩
- ⟨‘*Hunt Lowry*’, ‘ *Produced by*’, ‘*A Walk to Remember*’⟩
- ⟨‘*A Walk to Remember*’, ‘*is-a*’, ‘*Film*’⟩

we have to do several processing steps. First, the relations don’t follow specific design rules, so we use simple processing rules to remove special tokens and make the phrases lower cased. Some relations are directed in the opposite way (indicated by the ‘~’) symbol, which we reverted by changing subject and relation. In this particular example, the information that ‘The Secret Life of Bees’ is another ‘Film’ is missing. We added this information by using the database Freebase (Bollacker et al., 2007) from which the original triples are generated. After the processing, the new triples look as follows:

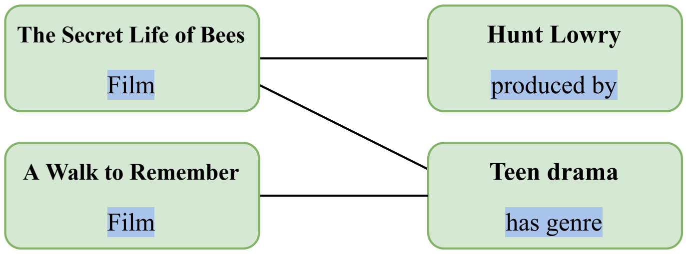

Figure 5.6.: A knowledge graph with 4 nodes (green). Each node has one type attribute (blue). For this graph, relations between nodes can be derived from the node type attribute.

- ⟨'*A Walk to Remember*', '*has genre*', '*Teen drama*'⟩
- ⟨'*The Secret Life of Bees*', '*has genre*', '*Teen drama*'⟩
- ⟨'*The Secret Life of Bees*', '*is-a*', '*Film*'⟩
- ⟨'*A Walk to Remember*', '*is-a*', '*Film*'⟩
- ⟨'*A Walk to Remember*', '*produced by*', '*Hunt Lowry*'⟩

We show the resulting knowledge graph in Figure 5.6. Each node has one node type attribute that is completely open-domain. Again, the relation between nodes can be derived by this node type attribute. In comparison to KOMODIS this property can be problematic. For simple graphs, an entity (for example 'Hunt Lowry') only needs one node type description (here 'produced by' which implies that this person produced anything that is connected to this node). However, if this node is used differently in a larger graph, we have to create another node of 'Hunt Lowry' with a different type attribute. For example, 'is-father' if we want to connect his children as nodes to the graph as well. This will be relevant in the next subsection, where we plan to make the graphs larger.

5.2.3. Sub-Graphs

The graphs from the previous section are precisely tailored to the dialogues, since we only used the related context information. The graphs contain exactly the facts and opinions that are required to generate the dialogue. At inference, when using a real information retrieval system, this will not be the case. Instead, graphs might have information that is irrelevant or important facts might even be missing. We see the former as a partially desired behavior. Having additional information to choose from gives the neural conversation model more freedom for an appropriate contribution. In this subsection, we expand all graphs with additional information to see if bigger graphs help the training process, to estimate space complexity of

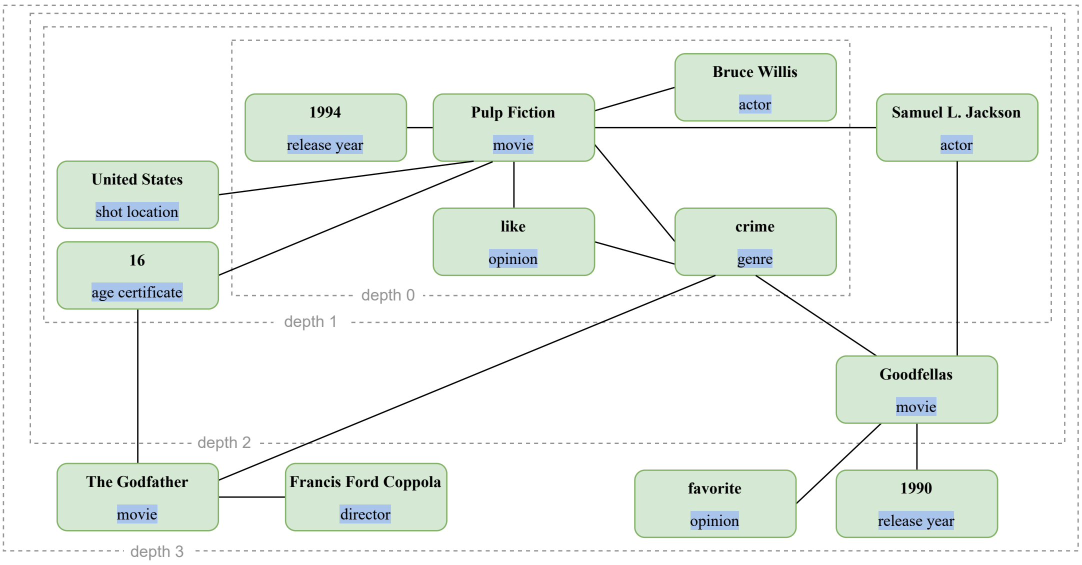

Figure 5.7.: Illustration of the knowledge graphs for a KOMODIS dialogue for different graph depths. Relations are undirected. Every node (green) has one type attribute (blue).

bigger graphs and to create a more realistic evaluation setup by providing more graph nodes than necessary. We separate this section for each dataset since their graphs require some individual processing.

KOMODIS To augment the graphs from KOMODIS we use IMDb again, as the original data was built with it. We define the base graph as *depth 0*. In a first step, we add all facts of the current movie that are used across the dialogues, except for the ones that we intentionally left out due to the *unknown* personality attribute. For all newly added entities which can be opinionated, we create a random opinion according to the opinion probability distribution introduced in the last chapter (except for *unkown*). We call this sub-graph *depth 1*. For these graphs the node distance of a new node to the already existing movie node is not greater than 1, except for new opinion nodes. In the next step, we augment them by allowing 2-hop distances, but with the restriction that we allow only one additional new movie. Additionally, we allow only up to three persons per movie and don't extend the graphs with new trivia facts. Otherwise, the graph would grow too fast. Again, we define opinions for all new entities. We call this version graphs *depth 2*. In a last step, we allow 3-hop distances from the original movie node and allow a total of three different movies with the same restrictions from the previous computation step.

depth	max hops	rules and constraints
0	0	Information from the speaker profiles only.
1	1	All information about the original movie based on the dataset.
2	2	Information from one additional movie. Maximum of 3 actors per movie. No additional trivia.
3	3	Any 3-hop information. Maximum of 3 actors per movie. No additional trivia. Based on depth 2.

Table 5.1.: Different graph depths for KOMODIS graphs with their constraints. Max hops is the maximum distance between a new node and the nodes from the base graphs.

Figure 5.7 shows an example graph with all different graph depths. The sub-graphs are incomplete to keep the figure clear. The shot location, age certificate and the actor ‘Samuel L. Jackson’ are not part of the speaker profile and are therefore only available in the *depth 1* graph. Here, everything has a 1-hop distance to the movie node. For *depth 2* the new movie ‘Goodfellas’ is added, which has a 2-hop distance to ‘Pulp Fiction’. While the movie ‘The Godfather’ also has a 2-hop distance to ‘Pulp Fiction’ it is only present in the *depth 3* graph, as the *depth 2* graph already contains a movie. ‘Francis Ford Coppola’ as the director of ‘The Godfather’, as well as the release year of ‘Goodfellas’ have 3-hop distances to the original movie. Table 5.1 summarizes the different graph depths and their creation restrictions.

OpenDialKG The OpenDialKG dataset has a per-utterance growing knowledge graph for each dialogue. Because we want to use these graphs as a static knowledge base, we always use the full graph (with all nodes) by default, which we call *depth 0*. Similar to *depth 0* from KOMODIS, this version of the graph only contains information that is used at some point during the conversation. For the bigger graphs, we added new nodes with 1-hop and 2-hop distances to that base graph. In order to keep the graphs in a manageable size (similar to the size of the KOMODIS graphs), we had to restrict the additional nodes to up to 3 for each existing node. Even with this restriction, the graphs for *depth 3* contain more than twice the number of nodes, compared to KOMODIS. Table 5.2 summarizes the different graph depths and their creation restrictions. Since the original graphs were created by accessing Freebase Bast et al. (2014), we also used that database to augment the graphs.

After the graph creation, we augment all dialogues for both datasets with all graphs and treat them as additional context like the original speaker profiles. This restricts all trainings to stick to one specific graph per epoch per dialogue, but the dataset size is sufficient, so that

depth	max hops	rules and constraints
0	0	Full graph from the dataset.
2	1	2-3 additional nodes for each node from the base graph, extracted from Freebase Bast et al. (2014).
3	2	2-3 additional nodes for each new node from depth 1, extracted from Freebase Bast et al. (2014).

Table 5.2.: Different graph depths for OpenDialKG graphs with their constraints. Max hops is the maximum distance between a new node and the nodes from the base graphs.

graph augmentation is not needed. Both dataset augmentations are available on GitHub.[1]

5.2.4. Context Length Requirements

To feed a knowledge graph into a Transformer model, we have to encode and transform the knowledge graph into sequences. This can be done by simple concatenation of the tokenized knowledge triples. We refer to this as the *base-triples* encoding approach. To estimate the space complexity of this encoding, we measure the correlation between the sequence length of the tokenized triples (as number of tokens) and graph size (as number of nodes). We show the results in Figure 5.8.

There is an almost linear correlation between number of graph nodes and number of context token. For KOMODIS we also have the paraphrased data available, as we have sentence patterns for all facts and opinion from the dataset collection procedure. We refer to the paraphrased context encoding with *base-paraphrased*. For KOMODIS the sequence length is marginally longer for paraphrased context, since that encoding needs more tokens than their triple equivalent. However, the difference is minor, and it seems that encoding triples concisely does not help to reduce the required space complexity significantly. This can only be achieved by reducing the actual content itself (without losing information), which we will describe in the next section.

5.3. Graph Attention Transformers

In this section, we propose our new and more concise way of encoding knowledge graphs for Transformer models. We first give an overview of our model in Subsection 5.3.1, then

[1] https://github.com/fabiangal/space-efficient-context-encoding-acl21

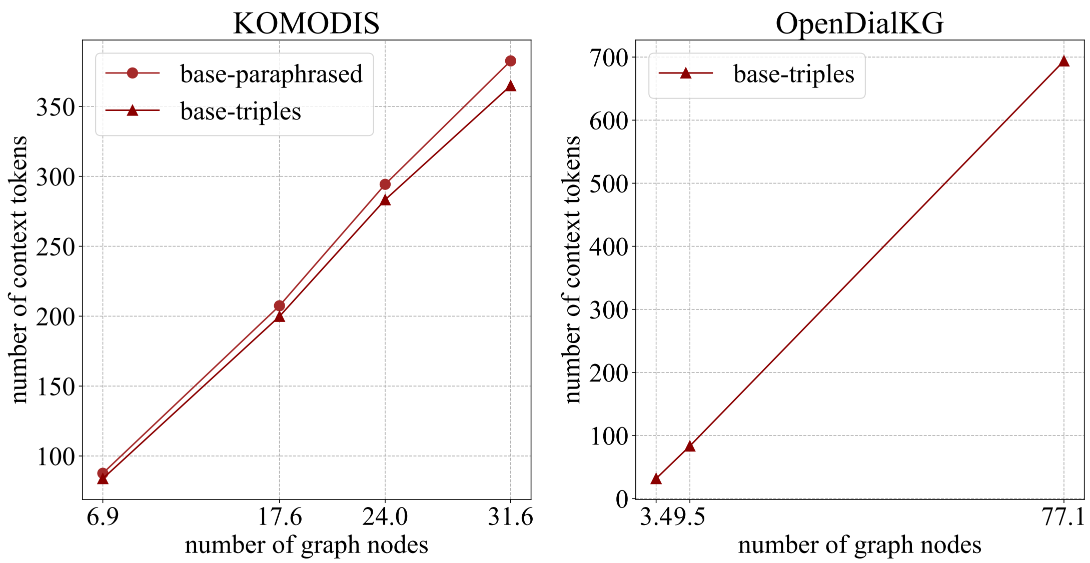

Figure 5.8.: Average number of context tokens in the input sequence for different graph sizes measured in number of graph nodes for KOMODIS and OpenDialKG. Data extracted from the whole train subset.

propose our encoding in Subsection 5.3.2 and show the idea of graph attention in Subsection 5.3.3.

5.3.1. Model Overview

For all experiments we use a GPT-2 model as our underlying language model. The model has 117 million parameters with 12 self-attention layer and 768 dimensional embeddings. Hyperparameter were mostly taken from the GPT-2 pre-training. We list all relevant parameter in Table 5.3.

Similar to our experiments with KOMODIS in Chapter 4 we use three layers of input embeddings. One for the word tokens, one for segment tokens and the pre-trained positional encodings. Encoded sub-graphs are concatenated to the left side of the sequence. At inference, a knowledge estimator creates a sub-graph based on the dialogue history. The full model architecture is visualized in Figure 5.9. At training the knowledge estimator is not needed, as we prepared all graphs beforehand.

Loss Function Similar to our KOMODIS experiments we optimize model weights θ by minimizing negative log likelihood for all x_i from a training sequence $\mathbf{x} = (x_1, x_2, x_3, ..., x_n)$

default hyper parameter	
batch size	32
optimizer	Adam
β_1	0.9
β_2	0.999
ε	1×10^{-8}
learning rate	6.25×10^{-6}
dropout	0.1
weight decay	0.01

Table 5.3.: Default hyperparameter settings for our experiments. If one of the parameters is not specified in an experiment, these values are used.

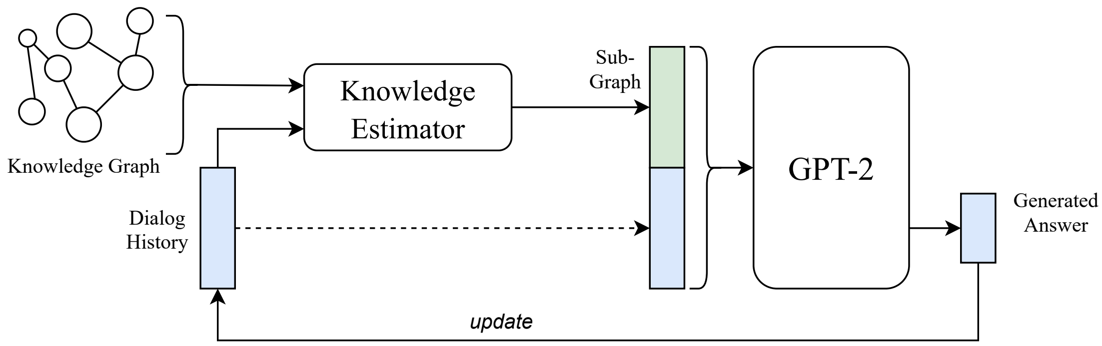

Figure 5.9.: Model architecture: A knowledge estimator creates a sub-graph based on the previous conversation. The processed sub-graph and input sequence are concatenated and fed into the GPT-2 decoder.

that are part of the last utterance (contribution) in that sequence:

$$\lambda_{\text{lm}} = \sum_i -\log P(x_i|x_{i-1}, x_{i-2}, ...; \theta). \tag{5.1}$$

This is a standard language modelling approach. This time we do not use the additional next-sentence classification as it greatly increases the required hardware resources and therefore training time, especially for the larger graphs.

Knowledge Estimator Knowledge estimation is not part of our research, and we therefore only implement a basic system to use the model at inference independently of the test dialogues. For KOMODIS we use simple string comparisons with a handcrafted set of movie titles and actors. As soon as one of these entities are detected, we create a graph with full

1-hop information about these entities. For OpenDialKG we use a similar approach and limit the search space to a handcrafted set of entities from music, films and books genres. If graphs become too large, we remove the oldest nodes. For evaluation, we decided to use graphs from the test set and stick to the training constraints: Only talking about a specific movie for KOMODIS and only one specific topic for OpenDialKG. This ensures that the quality of the knowledge estimator does not influence our results that focus on viability and efficiency on our graph encoding. We provide more details about our experiments and evaluation in future sections.

Sequence Lengths We use a maximum sequence length of 384 tokens. We have to set this limitation because of increasing hardware resource requirements for larger sequences. We reserve up to 128 tokens for the dialogue history and 256 tokens for the sub-graphs. These numbers are derived from the number of context tokens from our larger sub-graphs. More details will follow in the experiments section.

5.3.2. Knowledge Graph Encoding

The first step of our graph encoding is to transform the graphs into appropriate sequences. The *base-triples* and *base-paraphrased* encoding serves as our baseline. Paraphrased context is encoded exactly as in the previous KOMODIS experiments and for the triples we add three new segment tokens <*sub*>, <*rel*> and <*obj*> for subject, relation and object respectively. All triples are concatenated and added to the dialogue history. We use the following triples as an example:

1. ⟨'*Pulp Fiction*', '*release-year*', '*1994*'⟩
2. ⟨'*Pulp Fiction*', '*has-actor*', '*Bruce Willis*'⟩
3. ⟨'*Pulp Fiction*', '*is-a*', '*movie*'⟩

Figure 5.10 shows the related example encoding. To keep the illustration manageable, we only added triple 1 and 2, but triple 3 would have to be added in the same way. The following two encodings focus on transforming the real graphs into a sequence and are part of our new graph attention transformer idea.

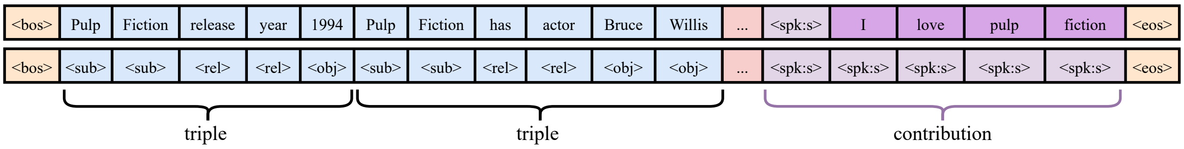

Figure 5.10.: Sequence encoding example with knowledge graph encoding (blue), implied dialogue history (red) and the contribution (purple) to be predicted. Word level (top row) contains the language vocabulary of the triples, and segment level (bottom row) separates the segments through new tokens. We refer to this as *base-triples*.

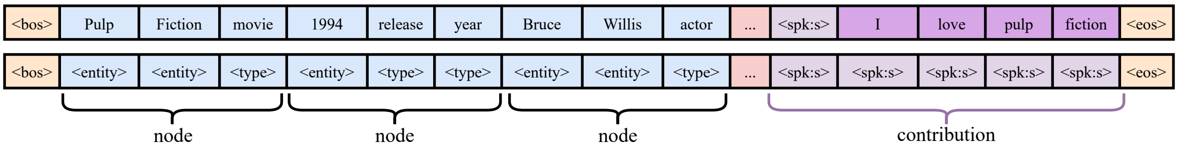

Figure 5.11.: Sequence encoding example with knowledge graph encoding (blue), implied dialogue history (red) and the contribution (purple) to be predicted. Word level (top row) contains the language vocabulary, and segment level (bottom row) separates the segments through new tokens. We refer to this as *series-encoding*.

Series Encoding For this encoding we use two new segment tokens: <*entity*> and <*type*>. Each node of a knowledge graph is encoded as an entity and type pair. The segment tokens are used to differentiate between these types. Following the example from the *base-triples* encoding, we need to encode three nodes: The movie node 'Pulp Fiction', the release year node '1994' and the actor node 'Bruce Willis'. We use the original word embeddings for the node types. Figure 5.11 shows the example described. This sequence is already significantly shorter, since the entity 'Pulp Fiction' needs to be encoded only once and the triple that defines 'Pulp Fiction' as a movie is given implicitly by the movie node. Note that this encoding loses the information about the relation between the entities. It is not clear, if the release year belongs to Pulp Fiction. To take that into account, we have to model the graph structure as well, as we will show in the next subsection.

Parallel Encoding In addition to the series encoding we also experiment with a more concise encoding utilizing the segments layer to store node type information. Instead of creating node and node type pairs in a series, we encode the node type information in the segments layer in parallel to the node encoding. Figure 5.12 shows this, following our previous example. The advantages of this encoding are even shorter sequences, no new segment type tokens

Figure 5.12.: Sequence encoding example with knowledge graph encoding (blue), implied dialogue history (red) and the contribution (purple) to be predicted. Word level (top row) contains the node content and segment level (bottom row) the related node type content. We refer to this as *parallel-encoding*.

and the node and its node type is encoded at the same sequence positions. The disadvantages, however, are that we have to add word token embeddings, which could cause issues when fine-tuning the model. Further, we have to make use of the padding token <*pad*>, since node content and node type content does not always have the same number of tokens.

Context Lengths To evaluate the sequence length improvements compared to the *base-paraphrased* and *base-triples* encodings, we compute the sequence lengths for the same graphs with our new encodings. Figure 5.13 shows the space complexity for all encodings. Number of context tokens increases linear for the new encodings as well, but with a significantly lower gradient. For bigger graphs, we achieved a reduction of up to 3.6 for KOMODIS and 3.5 for OpenDialKG. Graphs from OpenDialKG grow much faster due to the different structure of the dialogue context and the underlying knowledge graphs. The context graph for OpenDialKG is initially rather small (3.4 average nodes compared to 6.9 nodes from KOMODIS and much fewer tokens) and increases very fast with more hops. The KOMODIS context graph contains information about plot and trivia, which are normally longer strings that belong to one entity, thus the benefit of *series-encoding* and *parallel-encoding* regarding this information is rather small compared to the baselines. With increasing graph sizes, however, both approaches show similar behavior. Concluding, the sequence length reduction correlates with the average number of edges per node.
Our *series-encoding* requires 14% and 30% more context tokens on average compared to the *parallel-encoding*. Later experiments will reveal if the parallelization of word tokens is worth the space savings.

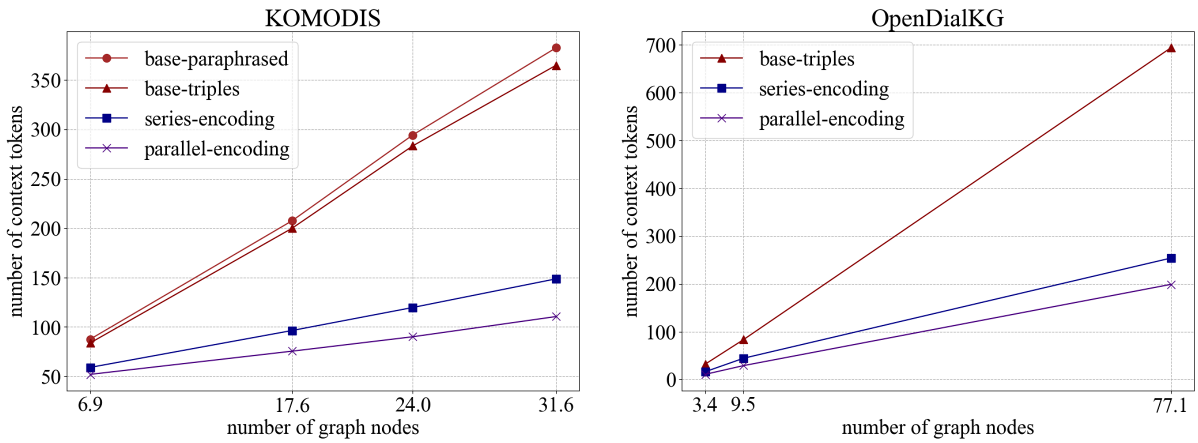

Figure 5.13.: Average number of context tokens in the input sequence for different graph sizes and encoding strategies measured in number of graph nodes for KOMODIS and OpenDialKG. Data extracted from the whole train subset.

5.3.3. Graph-based Attention

As already indicated in the previous section, with the new node-based encoding we lose the relations between entities. The nodes are encoded in the input sequence, but the information about the edges (i.e., which nodes are connected) is missing. To preserve this structure information, we make use of the general architecture of Transformer models. The GPT-2 model consists of 12 to 24 identical structured Self-Attention layers. Each layer is supposed to compute a hidden state for each input embedding based on the hidden states from the previous layer. For every hidden state h_i^l from layer l at the i'th position holds:

$$h_i^l = \sum_{j \in S} w_{ij} (\mathbf{V}_{\text{weights}}{}^{l-1} h_j^{l-1}), \tag{5.2}$$

where

$$w_{ij} = \text{softmax}_j (m_j + \mathbf{Q}_{\text{weights}}^{l-1} h_i^{l-1} \cdot \mathbf{K}_{\text{weights}}^{l-1} h_j^{l-1}) \tag{5.3}$$

is the weighted sum based on the previous layer $l-1$, a future mask m and learnable weights $\mathbf{K}_{\text{weights}}$, $\mathbf{Q}_{\text{weights}}$, and $\mathbf{V}_{\text{weights}}$. The authors call this Self-Attention, since the weighting for each feature j in h_i^l is based on its own values. If $m_j = 0$ for all j, every single h_i^{l-1} influences every single h_i^l. In our model each position represents one word token and therefore (without masking) we can visualize the GPT-2 layers as a fully connected word graph like in Figure 5.14. A word token would be represented by a node, and a hidden state is the new representation of a node based on all nodes from the previous computation step. This visu-

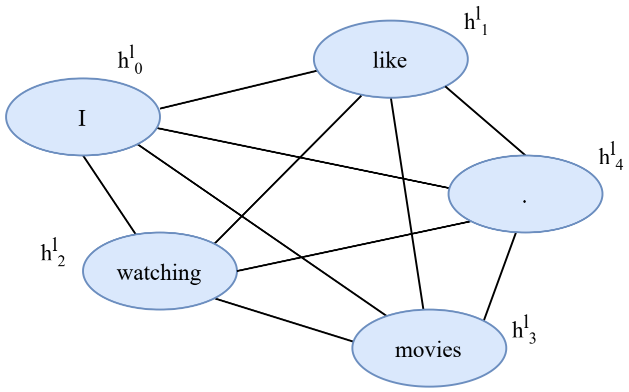

Figure 5.14.: Visualization of a Self-Attention Decoder without masking as a fully connected word graph for the input sequence: "I like watching movies."

alization shows the importance of additional effort to make the computations order-variant, as without, the word order of the sentence: "I like watching movies" is lost. The positional embeddings that are added to each node individually give the model a way to process these nodes as if they were in a specific order.

In addition to the positional embeddings a masking—what the authors call *future masking*—also causes order-variance. However, the goal of future masking is to prevent the model from seeing the future and therefore the label itself. The GPT-2 model was created with Generative Pre-Training (see Subsection 2.2.1), which means that the model was tasked to predict the next token of a given sequence. If the model can attend to future tokens, it would learn to copy the next token to the current output instead of attending the previous sequence. Future masking prevents exactly this by removing the information flow from all future tokens. With our new encoding the input sequences are a combination of real graph nodes (the context) and word tokens (the dialogue history) and in order to preserve the graph structure of the knowledge graphs, we manipulate the attention weights similar to the future masking. This resembles the message-passing approach of graph neural networks Gilmer et al. (2017).

Under the assumption that the Key and Query matrices at layer l for an input sequence of length l_{seq} with an embedding size of l_{emb} (ignoring Multi-Head Attention) are:

$$\mathbf{K} = \mathbf{h}^l \cdot \mathbf{K}^l_{\text{weights}} = \begin{bmatrix} k_{1,1} & k_{1,2} & \dots \\ \vdots & \ddots & \\ k_{l_{\text{seq}},1} & & k_{l_{\text{seq}},l_{\text{emb}}} \end{bmatrix},$$

$$\mathbf{Q} = \mathbf{h}^l \cdot \mathbf{Q}^l_{\text{weights}} = \begin{bmatrix} q_{1,1} & q_{1,2} & \dots & \\ \vdots & \ddots & & \\ q_{l_{\text{seq}},1} & & q_{l_{\text{seq}},l_{\text{emb}}} & \end{bmatrix}$$

The attention weights would be computed by multiplying $\mathbf{Q}$ and $\mathbf{K}^T$ with a row-wise Soft-Max, requiring a future mask $\mathbf{M}$ that is added element-wise (here indicated with $\oplus$) to remove information flow from the future:

$$\mathbf{Q} \cdot \mathbf{K}^T \oplus \mathbf{M} =$$

$$\begin{bmatrix} q_{1,1} & q_{1,2} & \dots & q_{l_{\text{seq}}} \\ \vdots & \ddots & & \\ & & & \\ q_{l_{\text{emb}},1} & & & q_{l_{\text{emb}},l_{\text{seq}}} \end{bmatrix} \cdot \begin{bmatrix} k_{1,1} & k_{1,2} & \dots & k_{l_{\text{emb}}} \\ \vdots & \ddots & & \\ & & & \\ k_{l_{\text{seq}},1} & & & k_{l_{\text{seq}},l_{\text{emb}}} \end{bmatrix} \oplus \begin{bmatrix} 0 & -\infty & \dots & -\infty \\ 0 & 0 & \dots & -\infty \\ 0 & \vdots & \ddots & -\infty \\ 0 & 0 & \dots & 0 \end{bmatrix}.$$

A value of $\mathbf{M}_{ab} = 0$ has no effect on the Soft-Max computation and therefore means *no masking* and a value of $\mathbf{M}_{ab} = -\infty$ means that the Soft-Max will convert this value into a 0, fully *masking* out this information source.

For subsequent considerations of our masks we will use a 1 for *no masking* and a 0 for *masking* in order to make the visualizations more intuitive. Also note that the above matrix computations are only one way to achieve Self-Attention, and other ways might be found in the literature, especially when considering multiple computation heads.

With our new masks, we have to map the following rules:

1. Having future masking for the dialogue history.
2. Expressing the graph structure through attention masking between graph nodes.
3. Allowing full attention from the dialogue history to the graph.
4. Masking out attention from the graph to the dialogue history.

Rule (1) and (2) are straightforward. We need to keep the current future masking for the dialogue history, as we still want to predict new tokens as usual and the graph part of the input sequence needs an attention mask that expresses the graph structure instead. Rule (3) is important since the dialogue generation is conditioned on the history and graph. However, the graph does not need to attend the history (rule (4)) as we want the graph embeddings to be independent and as close as possible to graph neural network approaches. We visualize the information flow in Figure 5.15. Arrows indicating the allowed flow of information from one hidden layer to the next one between each hidden state. Each hidden state is also allowed to attend itself, which we do not show in the Figure to keep it clear.

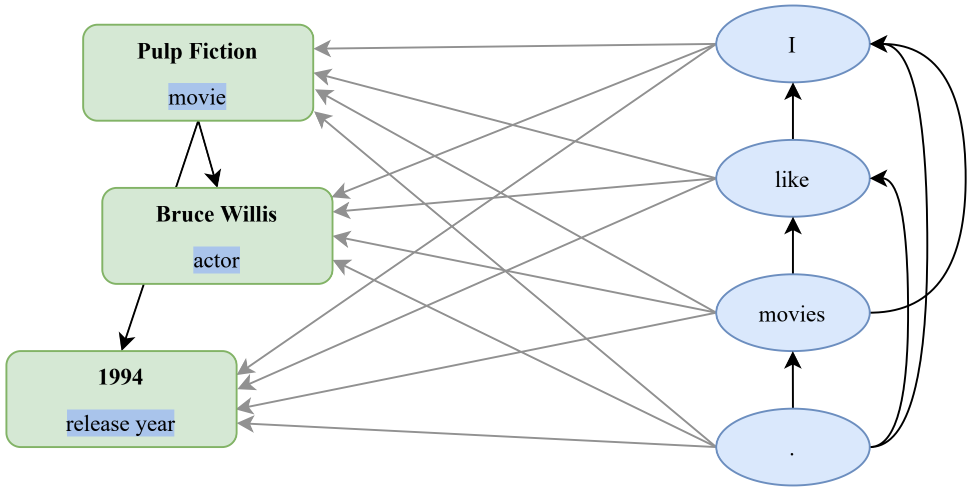

Figure 5.15.: Information flow graph for a knowledge graph and a dialogue history. Graph and history are shortened for better visualization. Arrows show allowed information flow.

According to the above matrix multiplications for inner Self-Attention, we need to map the four rules as shown in Figure 5.16. In the bottom right part (blue ones, red zeros) we have the usual future masking. In the top left part (green) we have the graph attention, that we will explain in the next paragraph. The top right part (yellow zeros) masks out the attention from the graph to the dialogue history. And the bottom left part (purple ones) maps the attention from the dialogue history to the knowledge graph. Bold ones indicate self attention.

Graph Attention Graph nodes in the input sequence are time-invariant, and the graphs are designed with undirected edges. Thus, every attention mask is symmetric. If one node consists of multiple word embeddings (like 'Pulp; Fiction') we create separate values for each embedding. We always allow full self-attention. We consider the matrix using the example node '1994': Self-attention is given through the *one* in row three, line three. We allow attention to and from the node embeddings 'Pulp' and 'Fiction' in line three, row one and two, as well as row three, line one and two. We mask out the attention to the node embeddings 'Bruce' and 'Willis' with *zeros* in line three, row four and five, as well as row three, line four and five.

	Pulp	Fiction	1994	Bruce	Willis	I	like	movies	.
Pulp	**1**	**1**	1	1	1	0	0	0	0
Fiction	**1**	**1**	1	1	1	0	0	0	0
1994	1	1	**1**	0	0	0	0	0	0
Bruce	1	1	0	**1**	**1**	0	0	0	0
Willis	1	1	0	**1**	**1**	0	0	0	0
I	1	1	1	1	1	**1**	0	0	0
like	1	1	1	1	1	1	**1**	0	0
movies	1	1	1	1	1	1	1	**1**	0
.	1	1	1	1	1	1	1	1	**1**

1/0 Graph encoding
1 Dialogue history self-attention
0 Future masking
0 Graph to history masking
1 Graph attention for dialogue history

Figure 5.16.: Attention mask for the graph from Figure 5.15. Sequence is simplified for readability. A zero means that the according attention is masked out.

5.4. Experiments

To evaluate our graph encoding approach we conduct four series of experiments focusing on the impact of randomness for dialogue generation in Subsection 5.4.1, verifying the importance of graph masking in Subsection 5.4.2, analyzing the performance between parallel and series encoding in Subsection 5.4.3 and the influence of different graph depths in Subsection 5.4.4.

5.4.1. Decoding Algorithms

Decoding strategies do have an impact on the quality of the results. To reduce parameter permutations, we do a smaller series of experiments first to estimate whether we want to use standard deterministic beam-search or a non-deterministic sampling algorithm. We use a pre-trained GPT-2 for all experiments with parameter specifications from Section 5.3.1 and train models with both datasets KOMODIS and OpenDialKG) with *base-triples* and *series-encoding* and graph *depth-1*. For all four models we generate next utterances with standard beam-search with 4 beams and simple 3-gram blocking and with top-k-sampling and $k = 4$. We evaluate the generated utterances with respect to knowledge precision and naturalness. Results are discussed in Section 5.6.

In this work, we emphasize the model's ability to integrate additional dialogue context correctly. Here, models with beam-search perform better. Thus, further experiments focus on beam-search only.

5.4.2. Effectiveness of Graph Attention Masking

Graph masking encodes the relationships between the entities. We hypothesize that dropping these relationships will lead to an information gap, particularly for bigger sub-graphs, due to more entities that are not represented (well) in the training data. We therefore conduct a series of experiments where we don't use the proposed graph masking. We discuss the results in Section 5.6.3.

5.4.3. Encoding Strategies

In order to evaluate the different encoding strategies, we plan to train three baseline models. Two GPT-2 models with parameter specifications from Section 5.3.1 for KOMODIS with *base-paraphrased* and *base-triples* encoding, as well as one model for OpenDialKG with *base-triples* encoding.
We then train models with *series-encoding* and *parallel-encoding* for both datasets. We evaluate on knowledge precision, opinion precision for KOMODIS specifically and for general naturalness, consistency and interestingness. We discuss these results in Section 5.6.

5.4.4. Graph Depths

To evaluate the influence of the amount of available context, we conduct a series of experiments with various graph depth. We fine-tune 3 GPT-2 models on KOMODIS data for *depth-0*, *depth-1* and *depth-2* with *parallel-encoding*. Results are discussed in Section 5.6.
To reduce the number of experiments, we only use *depth-1* and *depth-2* in combination with the different encoding strategies and both datasets in a second series of experiments because bigger graphs at training tend to improve knowledge precision.

5.5. Automated Evaluation, And Its Limits

As extensively discussed in Chapter 3, automated evaluation of the quality of neural conversation models lack adequate automatic metrics. In this section, we discuss which automated metrics are helpful and why precision and recall are problematic for our datasets. We further test the usefulness of our NLI model from Chapter 4.2.1 for KOMODIS models.

Perplexity All of our models are trained by minimizing negative log likelihood, which correlates with perplexity. We know that perplexity measures the syntactic structure of the dialogues and generated utterances (Serban et al., 2016). That is why we conclude, that fluency can be approximated conveniently by calculating perplexity on test. We do use perplexity as the stopping criteria for our training, but this metric does not help to evaluate consistency towards a given context. We therefore cannot use perplexity to evaluate our models at all. The only way perplexity helps here, is that measuring a high perplexity is an indicator for a general problem of the training setup, i.e., an indicator that the model cannot produce fluent contributions at all.

Information Retrieval Metrics Precision, recall and F1 score are common metrics for automated evaluation in the field of information retrieval. We, however, are interested in the ability of reproducing entities and relations from knowledge graphs. That means in our case, that evaluating precision and recall require precise co-reference resolution, parse tree annotation and question labelling (e.g., entities, relations, intent). An example where precision metrics fail is the following. Given the speaker profile:

1. ⟨‘*Pulp Fiction*’, ‘*has-genre*’, ‘*crime*’⟩
2. ⟨‘*comedy*’, ‘*opinion*’, ‘*I like*’⟩

an appropriate contribution to a conversation could be: “It is a crime movie, but I am more interested in comedy films.” Here, simple word matching algorithms (counting the occurrences of ´crime’ and ´comedy’) will fail because they don’t respect the meaning of the sentence. A similar sentence with the same entities is: “I like crime movies, but that one is a comedy film.” In this case, the contribution violates the whole speaker profile. In this example, you also have to resolve the co-reference of the first word ‘It’, as it could reference to a different movie as well. If one asks: “Do you like crime movies?” the algorithm must not count ‘crime’ at all. Especially non-goal-driven dialogues are too diverse to cover all cases with rules and exceptions.

experiment	correct		incorrect	
	facts	opinions	facts	opinions
KOMODIS dataset	0.85	0.76	0.10	0.20
KOMODIS series-encoding-d1	0.93	0.72	0.10	0.11

Table 5.4.: Estimated consistency from our NLI model from Section 4.5 on 50 randomly chosen dialogue samples, manually evaluated.

Moreover, more generic word-overlap metrics (such as ROUGE or BLEU) will fail for the same reasons. And they also have the problem that they need a gold target label to compare with, which is not possible in dialogue generation tasks, as many good replies exist that are appropriate given a history and context.

NLI as a Consistency Metric A possible way to evaluate consistency automatically would be to use the NLI models as described in Section 4.5. To evaluate consistency, each generated utterance has to be tested against each context item. In our case, against each fact or opinion that is given in the particular dialogue. To validate this process, we manually evaluated responses from 50 dialogues from the original dataset and on *series-encoding-d1*, one of our best performing models from this chapter. Table 5.4 lists the results. The NLI model is able to find 85% of consistent fact-uses and 76% of consistent opinion uses on the dataset. In 10% of the cases, the model detected a correct fact-use as contradiction, and in 20% of the cases for opinions respectively. While 85% is an acceptable performance for the classification task, it is not precise enough to serve as an automated evaluation metric for the experiments in this chapter. The performance on the trained model is higher for facts (93%) but slightly lower for opinions (72%). We assume that the trained model produces more simple fact utterances that are easier to detect automatically, while the opinion utterances stay similar.

Because none of the metrics discussed above are sufficient for consistency evaluation, we perform a human evaluation instead, which we elaborate in the next section.

5.6. Human Evaluation

To get valuable results, we conduct an extensive human evaluation for all of our experiments. We separate the evaluation in two phases. We first evaluate the experiments on encoding strategies and graph attention mask effectiveness and then, in a greater second batch, the encoding strategies in graph depths. We also separated the dialogue generation process from the evaluation itself because having the participants chat with the models would have raised the scope of the study for our resources. We also find it problematic to instruct so many participants on exactly what our models should be able to do and what their limits are. We want our evaluation to be within the limits of our models and make sure that we have dialogues about the given knowledge graphs. To ensure this, we carefully instructed four colleagues (who were not involved in the creation of the models and did not know what the innovation was) how to interact with our models. This way, they created 20 dialogues per experiment beforehand. We then use these dialogues for the greater human evaluation.

We describe the method of our human evaluation in Subsection 5.6.1 and present and discuss all our results from the different experiments in the following Subsections 5.6.3, 5.6.2, 5.6.4 and 5.6.5.

5.6.1. Method

The evaluation study was managed by researchers not involved in setting up the models and experiments.

Participants 20 participants not familiar with our research and the goals of the study took part in the human evaluation. 9 Participants were women, and 15 participants stated that they already have experience with various forms of chatbots. 13 participants were 18–35 years old, 4 were 36–50 years old, and 3 participants were older than 50 years.

Procedure All participants were instructed before and supervised during the study by a supervisor to ensure their understanding of the metrics. We prepared 20 individual questionnaires (shown in Appendix B) for each participant, covering all dialogues from the experiments explained above. The questionnaire contains a survey guide and a set of dialogue pairs to evaluate, as well as facts and opinions as context. The experiments for decoding and graph mask effectiveness were evaluated separately in a pre-evaluation to reduce the number of permutations for the other experiments. The evaluation itself consists of three parts:
(1) Marking utterances that either entail or contradict the dialogue context. Based on these annotations, we measure the model's knowledge retrieval ability as the ratio between entailing utterances and the sum of entailing and contradicting utterances (*precision*). For that, we provide dialogue and context in a matrix.
(2) Rating the dialogue regarding naturalness, consistency and interestingness by choosing

an agreement value on a 7-point Likert scale for the statements: “Person B sounds natural.”, “Person B sounds consistent.” and “Person B sounds interesting.”. Here, person B is always a model, Person A a human.
(3) Choosing the better dialogue (from the dialogue pair) by answering: “To which Person B would you prefer to talk?”. Additionally, the participants could briefly reason their decision.

Materials The questionnaire contains a survey guide and a set of dialogue pairs to evaluate. The survey guide consists of four pages with examples and explanations for the participants. The following excerpts are from the guide.
(1) General instructions: *Following, you are presented with two dialogues between Person A and Person B with according background information. The dialogues are completely independent of each other. You must read both dialogues carefully. Please take time for this task.*
(2) Instructions for evaluating the knowledge and opinion precision: *Please remember that the evaluation is for Person B only! Please add ‘entailment’ to the fields, when the utterance entails a specific fact or opinion. Please add ‘contradiction’ if an utterance contradicts a specific fact or opinion. Please leave all other fields empty.*
(3) Instructions for rating the dialogues on the 7-point Likert scale: *Please rate the three statements for each dialogue on a scale from 1 to 7, where 1 means that you strongly disagree with it and 7 that you strongly agree with it. Please rate all statements independently from the given facts and opinions. For instance, if a dialogue contains wrong facts, it still can sound very natural.*
(4) Instructions for deciding between two dialogues: *Please rate intuitively with which Person B you would prefer to talk. Please reason your decision briefly.* All instructions are provided with examples.

5.6.2. Decoding Algorithms

According to our human evaluation, beam-search reproduces knowledge with greater precision for all models compared to top-k-sampling. Results are shown in Table 5.5. Human annotators agreed with 69% to 74% of the model’s knowledge-based statements when using beam-search, while they agree with 45% to 56% of the statements when using top-k-sampling. The significant precision drop can be explained by the fact that top-k-sampling utilizes randomness, which contradicts the precise reproduction requirements for context-based dialogue generation. At each decoding step, the algorithm randomly samples one

experiment	knowledge precision		naturalness	
	base-triples	series-enc-d1	base-triples	series-enc-d1
KOMODIS beam-search	**0.69**	**0.74**	5.0 (1.5)	4.8 (1.6)
KOMODIS top-k-sampling	0.52	0.56	**5.9** (1.2)	**5.9** (1.3)
OPENDIALKG beam-search	**0.73**	**0.70**	4.0 (1.6)	3.4 (1.5)
OPENDIALKG top-k-sampling	0.54	0.45	**5.3** (1.4)	**5.4** (1.3)

Table 5.5.: Human evaluation results for beam-search and top-k-sampling, with respect to the correct reproduction of dialogue context. Precision as the ratio between entailing utterances and the sum of entailing and contradicting utterances. Naturalness on a 7-point Likert scale. Higher is better. Standard deviation in brackets.

of the k (we use $k = 4$) most probable tokens. In many cases, the conversation model ranks wrong entities high so that they are possible candidates for the random sampling. This shows that one core issue is the conversation model because it should never rank wrong candidates high. But language models learn statistical dependencies, and therefore it is not surprising that for example a wrong actor name is ranked higher than tokens that are not semantically appropriate. Training a model by minimizing negative log likelihood for next token prediction is binary: At each step, the algorithm tests for the appropriate token and punishes every other token equally, even if other tokens might make sense as well. Beam-search, on the other hand, produces more consistent utterances by strictly using the highest cumulative probabilities. This shows that our models learned to be consistent, as the correct entity has the highest probability in most cases.

Dialogues generated with top-k-sampling are considered more natural, less self-contradicting, and less repetitive. One issue with beam-search is repetition. Even though the added N-gram filtering does reduce repetition, we were unable to completely avoid it without a heavy length penalty. But a high length penalty (punishing short sequences to align them with probabilities of long sequences) causes short and generic contributions. For all other experiments, we therefore use a parametrization (see Subsection 5.3.1) that balance out both factors.

In this work, we emphasize the model's ability to integrate additional dialogue context correctly. Here, models with beam-search perform better. Thus, our further evaluation focuses on beam-search. For future experiments we consider advanced decoding algorithms such as nucleus sampling (Holtzman et al., 2020) or dynamic length beam-search (Roller et al., 2021) where the minimum sequence length is estimated by an additional neural network to improve the overall neural conversation model quality.

5.6.3. Effectiveness of Graph Attention Masking

Removing the attention masks for the knowledge graphs significantly reduces the knowledge precision for both datasets. We list results in Table 5.6. This effect is higher for bigger graphs. This behavior was expected, since the attention masks contain structural information about the nodes. Without the masks, the model cannot estimate the relation between entities. This is less problematic for smaller graphs, as the number of entities from the same type is small. For example, if there is only one actor in the graph, it is not necessary to draw the relation to the movie because the model can learn to use that actor regardless of the connection. Bigger graphs increase the amount of wrong entities from the same type, which is problematic without relation information. At training, the error signal is not clear and the model cannot learn the right decisions, except if there are other correlations in the data. But according to our results, this is not the case.

The drop in knowledge precision is lower for OpenDialKG. Since OpenDialKG is open-domain, the average number of different same-type entities is lower compared to KOMODIS. This aligns with our previous explanations. The importance of having relation information increases with the number of similar entities. Interestingly, for OpenDialKG, the naturalness does not decrease when removing the attention masks. We assume that this is because of the lower naturalness for the baseline models. Contribution with wrong facts might not hurt naturalness, if the dialogues are not very natural in the first place.

The evaluation validates our hypothesis that attention masks are important and that the models have to rely on them in order to contribute precisely, especially when the provided graphs are bigger.

5.6.4. Encoding Strategies

The results for models trained on KOMODIS are listed in Table 5.7 and for models trained on OpenDialKG in Table 5.8. We notice two major differences between the datasets. The agreements for naturalness, consistency and interestingness are worse for OpenDialKG with a drop of up to 1 point on a 7-point Likert scale. The second difference is that *series-encoding* works better for KOMODIS, while *parallel-encoding* causes higher knowledge precision for OpenDialKG. We elaborate both further in the following paragraphs.

experiment	precision knowledge	precision opinion	naturalness
KOMODIS series-enc-d1 no mask	0.61	**0.46**	4.1 (1.4)
KOMODIS series-enc-d1	**0.74**	0.36	**4.8** (1.7)
KOMODIS series-enc-d2 no mask	0.44	0.25	4.4 (1.4)
KOMODIS series-enc-d2	**0.73**	**0.43**	**4.8** (1.7)
OPENDIALKG series-enc-d1 no mask	0.54	—	3.9 (1.8)
OPENDIALKG series-enc-d1	**0.62**	—	3.9 (1.9)
OPENDIALKG series-enc-d2 no mask	0.37	—	**3.8** (1.9)
OPENDIALKG series-enc-d2	**0.46**	—	3.7 (1.7)

Table 5.6.: Human evaluation results for models without graph attention masking. There are no opinions in the case of OPENDIALKG. Knowledge and opinions as precision (ratio between entailing utterances and the sum of entailing and contradicting utterances). Naturalness on a 7-point Likert scale. Higher is better. Standard deviation in brackets.

Baseline models have the lowest perplexity for both datasets. This effect is bigger for OpenDialKG with a perplexity difference of 1.5 between the baseline and the best model *parallel-enc-d1* regarding perplexity. Our new encodings, especially the graph attention masks, require significant re-learning of model weights, as our encodings differ to a greater degree from the pre-training material. We assume that this causes higher perplexity, i.e., a flatter probability distribution over the vocabulary. Since perplexity is already low across all models, we don't classify the increase as a negative performance indicator at all.

The *series-enc-d1* models for both datasets have the highest win ratio across all models. Participants choose the series encoding models in two-thirds of all cases as the better dialogue (see Section 5.6.1 for the details of this metric). Due to the limited number of dialogues, we were unable to estimate the win-ratio between each model pair, but only the average win ratio. The evaluation result indicates that models trained with our series encoding produce dialogues that seem more engaging for participants compared to the baselines, but also compared to the parallel encoding. Parallel encoding saves additional space but also utilizes the segment layer for real word token embeddings, resulting in the final embeddings being summed up from two different word tokens. This is different from the pre-training setup, and the drop in win-ratio compared to series encoding is an indicator that adding word tokens together reduces training performance.

Series encoding and parallel encoding models trained on KOMODIS reach the same knowledge precision values as the baseline with 74% and 72% respectively. Contrary to our as-

experiment	ppl	win ratio (%)	precision knowledge	precision opinions	agreements natural	agreements consistent	agreements interesting
base-paraphrased	10.3	12.5	**0.74**	0.50	4.7 (1.7)	4.1 (1.7)	4.2 (1.2)
base-triples	**9.73**	43.8	0.69	**0.71**	**5.0 (1.5)**	4.0 (2.0)	4.6 (1.1)
series-enc-d1	10.01	**66.7**	**0.74**	0.36	4.8 (1.7)	4.5 (1.9)	**4.9 (1.1)**
series-enc-d2	10.28	62.5	0.73	0.43	4.8 (1.7)	4.2 (1.6)	4.4 (1.2)
parallel-enc-d0	9.94	28.6	0.56	0.42	4.5 (1.6)	4.4 (1.2)	4.4 (1.2)
parallel-enc-d1	10.07	56.3	0.70	0.33	4.5 (1.7)	4.5 (1.2)	4.5 (1.1)
parallel-enc-d2	10.36	60.0	0.72	0.57	4.8 (1.5)	**4.6 (1.5)**	4.5 (1.2)

Table 5.7.: Perplexity on the test set (lower is better) and human evaluation results for models trained on KOMODIS. Metrics explained in Subsection 5.6.1. Agreements are on a 7-point Likert scale (higher is better). Standard deviation in brackets. “base-*" are the baseline models; “series/parallel-enc-*" denotes the way the knowledge is encoded and “-*d0/d1/d2" is the depth of the graphs.

sumption, the graph encoding strategies do not help to increase the knowledge precision significantly, i.e., the consistency towards facts, compared to baseline approaches. Reaching the same level of knowledge precision with greatly reduced space complexity is a good improvement. We think that a pre-training tailored towards graph embeddings can improve knowledge precision further. The GPT-2 model was pre-trained with Future Masking only, and manipulating the masks differently creates a significant difference compared to the original training. We elaborate this further in the next Section. The opinion precision is lower in general, indicating that the task is harder to train. We notice a drop—on average—when using series or parallel encoding.

5.6.5. Graph Depths

According to the results from Table 5.7, graph depth correlates with knowledge precision, naturalness and win-ratio. Only the opinion precision has an outlier for depth 1. A bigger sub-graph leads to more difficult training data, as the model has more options to choose from. If a graph, for example, only contains one release year, the model does not need to learn the node connection from the movie to that year. Instead, it only needs to learn to use that specific year node. This can cause generalization issues in the learning process. Having multiple release years require the model to *understand* the structure of the graphs. Bigger graphs also allow more error tolerance for the knowledge retrieval component, as the estimation algorithm can be more greedy in the selection process.

experiment	ppl	win ratio (%)	precision knowledge	natural	agreements consistent	interesting
base-triples	**8.40**	65.0	**0.73**	**4.0 (1.6)**	3.9 (1.6)	3.6 (1.6)
series-enc-d1	9.93	**66.7**	0.62	3.9 (1.9)	**4.1 (1.9)**	3.5 (1.9)
series-enc-d2	10.53	51.3	0.46	3.7 (1.7)	4.0 (2.0)	**3.8 (1.6)**
parallel-enc-d1	9.88	38.5	0.70	3.4 (1.6)	3.2 (1.8)	3.0 (1.3)
parallel-enc-d2	10.44	32.5	0.62	3.4 (1.9)	3.6 (1.9)	3.3 (1.6)

Table 5.8.: Perplexity on the test set (lower is better) and human evaluation results for models trained on OpenDialKG. Metrics explained in Subsection 5.6.1. Agreements are on a 7-point Likert scale (higher is better). Standard deviation in brackets. “base-*" are the baseline models; “series/parallel-enc-*" denotes the way the knowledge is encoded and “-*d0/d1/d2" is the depth of the graphs.

Interestingly, we could not reproduce the correlation between graph depth and performance for OpenDialKG (see Table 5.8). As mentioned earlier, this dataset was created for graph-generation based on dialogues and not vice versa. This causes a different dialogue structure: Due to the recommendation task of the data collection, most entities in these dialogues (e.g., persons, books, movies) are exchangeable and not essential for the dialogue consistency. “Can you recommend me a crime book similar to X?”, for example, does not require ‘X’ to be very specific, of even unique. It only has to be a crime book. Expanding these knowledge graphs by adding 1-hop nodes, often adds additional valid entities that would also produce a consistent dialogue. But adding valid entities makes the training process harder without gaining any generalization for the model. Therefore, the OpenDialKG graphs need to be expanded by adding entities that don’t make sense in the conversation, to emphasize the model to choose the correct ones. This, however, requires more effort on the graph creation process. We leave this as an interesting next step for future research.

5.7. Conclusion

Using a knowledge base and personality descriptions are important elements for creating a human-like neural conversation model. A lot of research tries to find good models that incorporate the context well (see survey in Section 3.4.3). In this chapter, we proposed a new and concise sequence encoding for knowledge graphs, that enable Transformer-based sequence-to-sequence architecture models to incorporate graph context efficiently. Our encoding reduces the required sequence length by up to a factor of 3.6 compared to the baseline, which is simple concatenation of all information triples. The space reduction does not have any negative effects on the consistency of the models, as proven in the human evaluation. Having access to more context while generating an appropriate answer is beneficial and reduces the importance of knowledge retrieval components in the overall setup: The retrieval components need to precisely provide the required content to produce the next utterance. That makes the context choice of these components crucial, as it determines the content of the next utterance beforehand. Having more space for context in the decoder shifts the decision weight of the content towards the direction of the language model, as it has more information to choose from.

As already mentioned above, even our best model could not outperform the baseline in terms of knowledge precision. A concise encoding should allow Transformer-based models to efficiently learn consistency towards facts and opinions, though. One limiting factor in our experiments is the pre-training. All models are pre-trained only on language modelling tasks and fine-tuning these models similarly, like in our baseline approach, utilizes that better. For future research, it would be interesting to see how good our encoding works, when pre-trained on big corpora that already utilize our context encoding. This, however, requires large data sources with text based on structured knowledge. Unfortunately, to this point, there are no such datasets available. Another possible way to pre-train our encoding could be an auto-encoder approach: This would remove the need to have structured knowledge and text pairs, as it on requires the structured knowledge. The task would be to encode and decode the knowledge unsupervised, and then only use the (pre-trained) encoder for further dialogue training.

Evaluation in this area is difficult and getting valuable results requires extensive human evaluation. NLI models might be a good candidate for automated evaluation, but we were unable to produce models with sufficiently high performance yet, although our models have been trained on specifically tailored data. We still think that this field is promising for future research. Progress in this area does not only help to develop automated metrics, but also enables new ways of model training, such as reinforcement learning, which we discuss in the next chapter. Instead of collecting larger dialogue datasets and training bigger models with the goal to solve the generalisation problem of conversation with one model, these approaches focus on understanding the different necessary properties of conversations.

6. Improving Consistency with Reinforcement Learning

Reinforcement Learning opens up new possibilities to train neural conversation models towards consistent contributions by allowing various non-differentiable reward functions, as done by Mesgar et al. (2021). In this chapter, we investigate the potential of the REINFORCE algorithm for knowledge and personality consistency. For that, we make use of our NLI model from Section 4.5.5 as a reward function for a neural conversation model pre-trained on KOMODIS. These experiments are based on the research from Mesgar et al. (2021) with the PersonaChat dataset. We investigate that approach, reuse their models and data to compare the results to our experiments with KOMODIS, and evaluate the similarities and differences.

First, we discuss the limitations of next token prediction tasks in Section 6.1, then go over the specific reward functions that we will use in Section 6.2 and then describe our experiments and results in Sections 6.3 and 6.4.

6.1. Limitations of Next Token Prediction

Like many other natural language processing tasks, neural conversation models are trained by minimizing the negative log likelihood for each individual token on every response (see Section 2.2.1). This is common because stochastic gradient descent algorithms require a differentiable loss function and language generation can be seen as an iterative classification problem: Given an unfinished sequence of tokens, we choose the next token (from our vocabulary). Then, we repeat this step until our generated sequence is complete. However, that process of next token prediction has some disadvantages.

At training, this approach uses the correct next utterance as ground truth, which is not available at inference. Just using the most probable token at each decoding step (called greedy decoding) leads to error accumulation, since the model does not take the probabilities of future tokens into account. Different decoding strategies like beam search or sampling (see Chapter 3) are required to find sequences with how cumulative probabilities. However, deciding on these cumulative probabilities causes a few issues: Sequences of different lengths are not comparable, as cumulative probabilities become smaller with increasing multiplicative elements (because a probability is a number between zero and one), making long sequences less probable than shorter ones innately. Further, Holtzman et al. (2020) showed that human-generated text has a high probability variance, proving that cumulative probability does not fully correlate with generation appropriateness.

Another issue with next token prediction comes with the linearity of the ground truth itself: Usually, a classification problem is definite, which means that there is only one correct class for each sample. But a conversation can have many good continuations, which is not considered here at all. Unlikelihood training (see Section 3.4.1) tries to tackle that, but cannot eliminate the issue, as it only uses a finite set of additional examples in the same way.

One promising alternative to next token prediction is reinforcement learning. It enables the model to take future tokens into account and also don't restrict the generation to a ground truth. However, this approach requires precise automated reward functions, and has to deal with the common high data sparsity that comes with natural language.

6.2. Reward Functions

Model weights are optimized based on the results from the reward functions. Thus, making their quality crucial in the reinforcement learning setup. The model cannot learn more appropriate contributions than what the reward functions are capable of. Further, the reinforcement learning algorithm might reveal decoding strategies that maximize the reward values, but are far from the expected results. Mesgar et al. (2021), for example, reported high keyword repetition to trick a specific reward function to always maximize the reward value. Therefore, it is important to choose a balanced set of different well-working metrics to create a robust reward that covers all required aspects for evaluating an appropriate contribution from neural conversation models. For our experiments, we use the following three reward functions in various combinations.

6.2.1. Consistency Reward

To learn consistency, we select an NLI-based consistency reward for the KOMODIS experiments from Section 4.5. We use the estimated probabilities between generated utterances and context information directly as rewards, similar to Mesgar et al. (2021). The consistency reward $R_{\text{consistency}}$ for a generated utterance r with the true context $\mathbf{C}$ is computed by:

$$R_{\text{consistency}} = \frac{k_{\text{entail}}}{|\mathbf{C}|} \sum_{c \in \mathbf{C}} P_{\text{entail}}(c, r) - \frac{k_{\text{contr}}}{|\mathbf{C}|} \sum_{c \in \mathbf{C}} P_{\text{contr}}(c, r), \tag{6.1}$$

with k_{entail} and k_{contr} being coefficients to balance the influence of entailments and contradictions, and P_* being the model probability for an entailment or a contradiction between r and any c.

The output of the NLI model is computed by a standard feed-forward layer with soft-max activation, which does not produce real probabilities for our entailment predictions. Thus, the above reward signal does not necessarily correlate with the degree of entailment or contradiction. This causes a noisy reward signal–especially for lower values–and we decided to stabilize the rewards by only considering entailment or contradiction probabilities above 0.5:

$$P_* = \begin{cases} P_* & P_* \geq 0.5, \\ 0 & \text{otherwise.} \end{cases} \tag{6.2}$$

We do not differentiate between facts and opinions for the reward computations in our experiments. Since the model needs fully generated sentences, this reward can only be computed after the decoding process. We set the reward to 0 for every previous time step.

6.2.2. Language Modelling Reward

The consistency reward does not consider syntactic or semantic properties of the generated content. As we will show in the experiments, the consistency reward itself leads to contributions that only maximize that reward, producing text with keywords that force the NLI model to predict high entailment probabilities.

Therefore, we add a language modelling reward based on the negative log likelihood of a language model fine-tuned on KOMODIS, similar to Mesgar et al. (2021). The purpose of this reward is to maintain high fluency throughout the training, and the reward for a generated contribution r is computed by:

$$R_{\text{fluency}} = \frac{\alpha - \text{NLL}(r)}{\alpha}, \tag{6.3}$$

with α being a scaling factor to normalize the negative log likelihood to values between 0 and 1. We computed α by finding the highest log likelihood on our KOMODIS test set $\mathbf{R}_{\text{test}}$:

$$\max_{r_{\text{test}} \in \mathbf{R}_{\text{test}}} \text{NLL}(r_{\text{test}}) \approx 8.3. \tag{6.4}$$

For this reward, we specifically trained a GPT-2 model only on our KOMODIS dialogues not using the context to avoid correlations between the consistency reward and the language modelling reward. Therefore, the purpose of this language model is to only analyze fluency, making the additional context irrelevant for this reward.

6.2.3. Repetition Reward

Unnecessary repetition is a common decoding problem and can be tackled easier with automated reward functions compared to the previous ones. Similar to Mesgar et al. (2021), we count all repeated tokens in each contribution r and measure the proportion of not repeating ones:

$$R_{\text{repetition}} = 1 - \frac{\text{number of repeated tokens}}{\text{number of tokens}}. \tag{6.5}$$

This reward could be computed before the fully generated sequence is known. To align the process with the other metrics, we decided for this metric to wait until the end of the sequence as well.

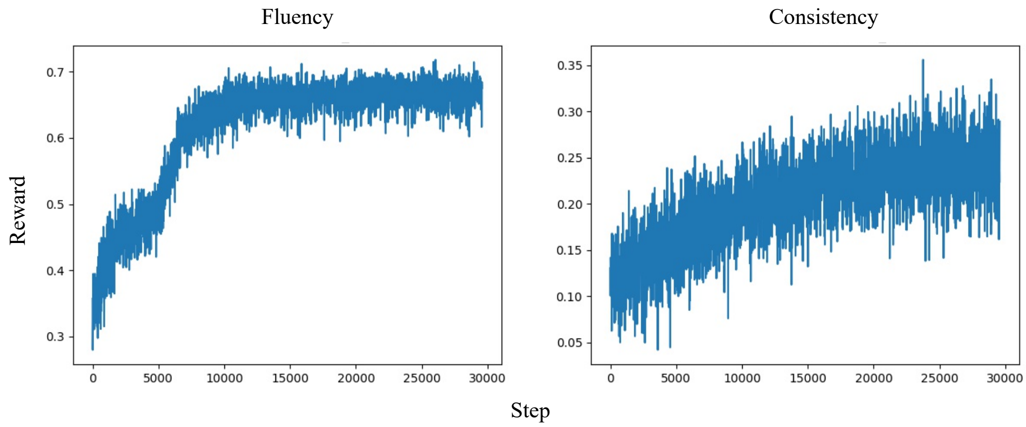

Figure 6.1.: Reward values for our reinforcement learning experiments on the PersonaChat data, only trained on the fluency reward (left) or the consistency reward (right).

6.2.4. Total Reward

All rewards are normalized between 0 and 1. Therefore, we can compute the total reward by simply adding them to a weighted sum:

$$R_{\text{total}} = f_{\text{c}} \cdot R_{\text{consistency}} + f_{\text{f}} \cdot R_{\text{fluency}} + f_{\text{r}} \cdot R_{\text{repetition}}, \tag{6.6}$$

with $f_{\text{c}} + f_{\text{f}} + f_{\text{r}} = 1$.

6.3. Experiments

We conduct all experiments on a reinforcement learning setup, using an Actor-Critic policy-gradient method. The pre-trained GPT-2 model serves as our agent. The critic model is a linear one-layer model that is tasked to estimate the rewards. The predicted value is then subtracted from the real reward, like in Mesgar et al. (2021), to reduce the variance in the estimated gradient. The agent creates responses with top-k-sampling and $k = 4$ with a maximum sequence length of 32. We only optimize the parameters from the linear language modelling layers of the GPT-2 models. We optimize with stochastic gradient descent with a learning rate of $l = 6.25 \cdot 10^{-5}$ and a batch size of 18. These values are taken from the previous fine-tuning step. We did not optimize these hyperparameters on an additional series of experiments. We trained our models on KOMODIS and PersonaChat in various reward combinations to explore the influence and effectiveness of these rewards.

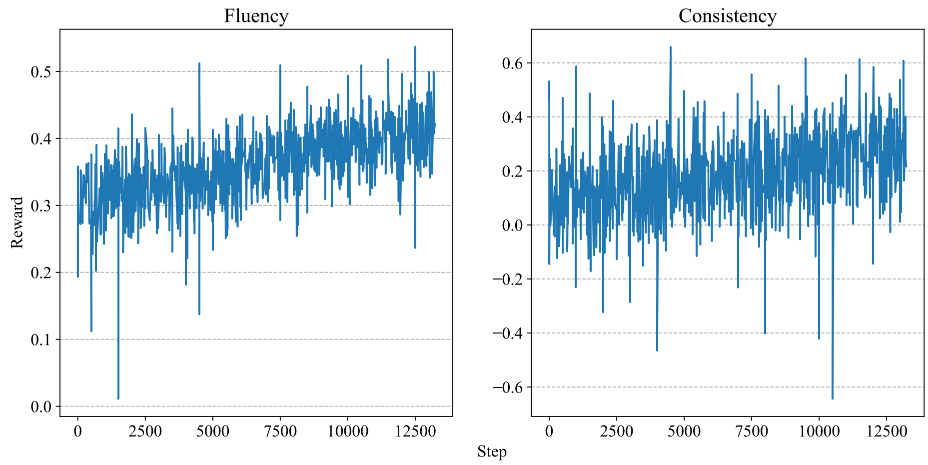

Figure 6.2.: Reward values for our reinforcement learning experiments on the KOMODIS data, only trained on the fluency reward (left) or the consistency reward (right).

For both datasets, we choose $f_\mathrm{c} = 0.4$, $f_\mathrm{f} = 0.3$ and $f_\mathrm{r} = 0.3$ when combining all rewards for one epoch. Thereafter, we repeated the training for all possible combinations with two ($f_1 = f_2 = 0.5$) and only one reward function ($f = 1$). In another series of experiments we increased the weight for the consistency reward to $f_\mathrm{c} = 0.7$ and $f_\mathrm{c} = 0.8$ when combined with either fluency or repetition reward.

6.4. Results

Figures 6.1 and 6.2 show the estimated rewards during training for both datasets, when only trained on one reward. For the PersonaChat data, fluency doubles from 0.35 within the first 10k steps and then does not increase further. This aligns well with the observations on the actual generated sequences (see Appendix C for generated sequences): Many sequences end up being a question about the partner's hobbies, e.g., "*that is cool. do you have any hobbies?*" This seems to be a very generic and therefore fluent reply in many cases. Only using the fluency reward overfits the model and shows the limitations of the fluency reward. The consistency rewards grows slower from 0.1 to 0.2 over the full training process and with higher fluctuations. This indicates that consistency is harder to learn. The increased fluctuations could also come from the consistency estimations, as that neural network doesn't have a 100% accuracy. The observations on the actual generated sequences show that sequences later in the training focus more on the persona descriptions and often copy them in a simpler way. We have added examples in Appendix C.

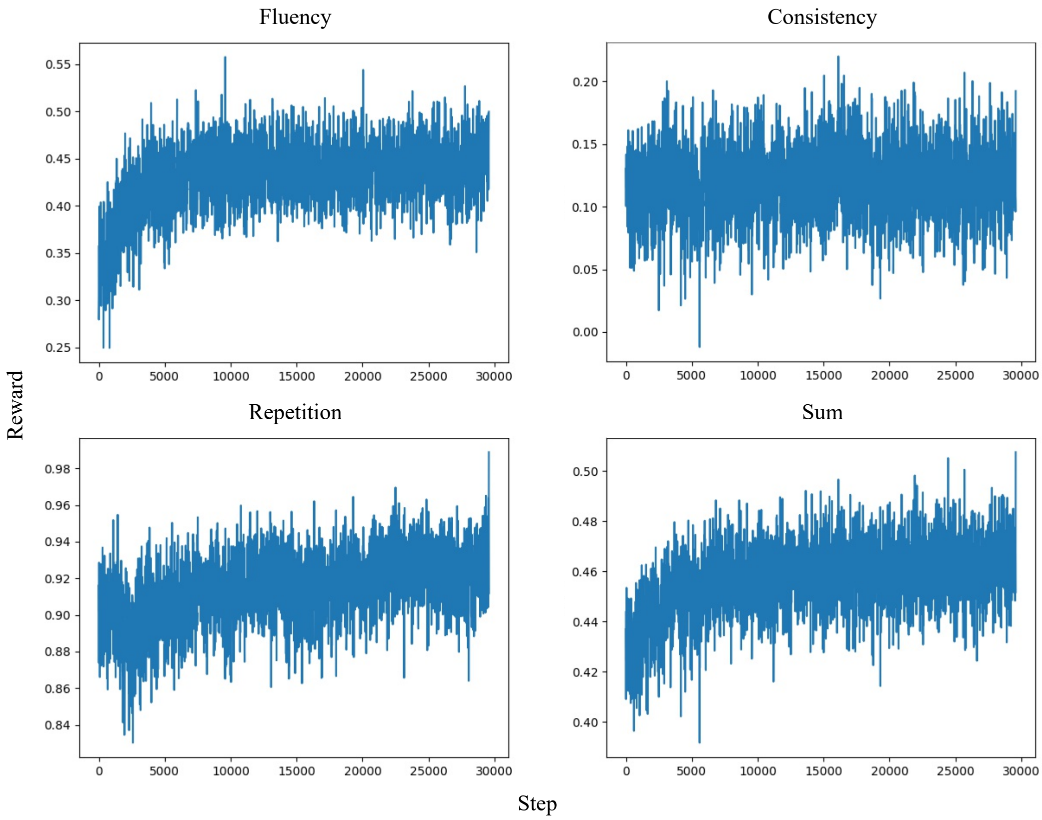

Figure 6.3.: Reward values for our reinforcement learning experiment on the PersonaChat data with all rewards being active.

The rewards for KOMODIS look similar, but the fluency reward increases significantly slower from 0.3 to 0.4 and fluctuations are high. Consistency increases from 0.1 to around 0.3, but has very high fluctuations as well. We investigated the generated sequences, but did not find significant changes between the original and trained model. For consistency, generated sequences from later training stages seem to be less contradicting. We conclude that the KOMODIS task is harder to train, at least for the given metrics. Here, the non-optimal NLI model is one reason for the high fluctuation in the consistency reward.

Figures 6.3 and 6.4 show the estimated rewards for repetition, fluency and consistency for both models, in a training setup that combined them to one reward signal. For PersonaChat the fluency reward increases only up to between 0.40 and 0.45, which was expected, as the other rewards reduce the overfitting on fluency in that training. However, the consistency does not increase significantly over the training. When investigating the generated sequences, we did not notice qualitative improvements on consistency as well. KOMODIS produces similar results, but here the fluency reward also does not increase during training. We interpret these results as follows: Learning fluency or consistency is a complicated task

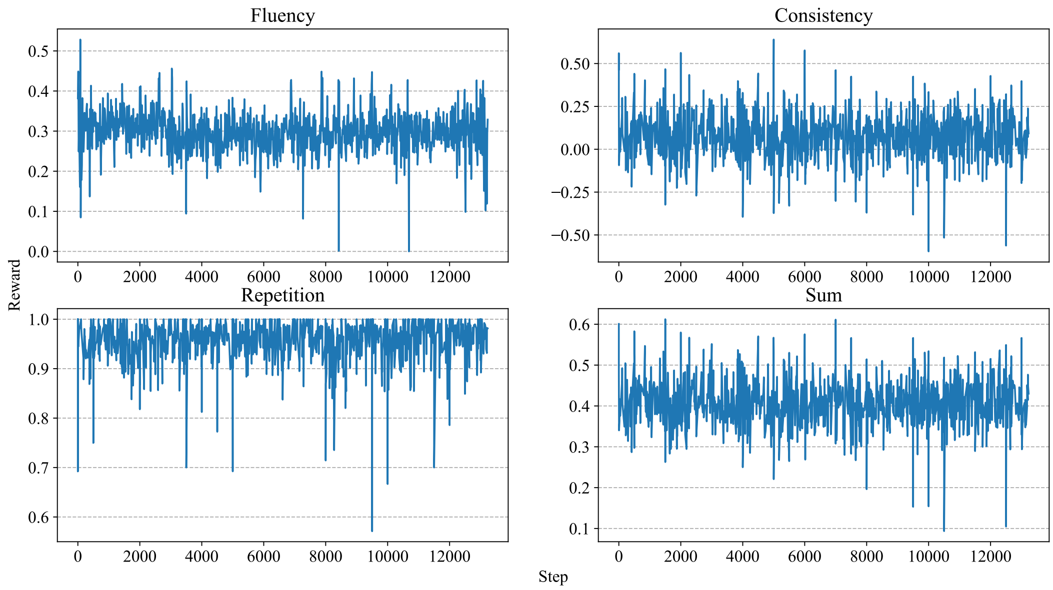

Figure 6.4.: Reward values for our reinforcement learning experiment on the KOMODIS data with all rewards being active.

by itself, and mediocre metrics to estimate a reward makes that task even more difficult. Training on a specific reward seems to work, but causes fast overfitting, as the reward function can be exploited by the models easily (see Mesgar et al. (2021) as well), for example by generating keywords that trigger a high consistency reward. Combining rewards seems promising, as this should make sure, that the model cannot overfit to one specific reward, but the model has to learn to produce sequences with high rewards for all metrics. This, however, does not work well in our setup. Especially for KOMODIS, the different reward signals seem to hurt each other so that the model does not learn anything at all. We assume that the combined task is too difficult to learn because the search space is too large. Having too many actions at each time step is problematic in reinforcement learning. One bottleneck in our architecture is the reward combination: We sum up the rewards according to Equation 6.6, which makes it impossible to determine the cause of a high or low reward. In standard stochastic gradient descent algorithms, this problem does not exist, as the weight optimization uses the derivations of the full loss function, which is not the case in reinforcement learning.

The repetition reward differs from the other rewards. Unlike the neural network-based rewards, the repetition reward is precise, as it measures the number of repeated tokens. Therefore, there is no noise in the reward value because of wrong classifications. This produces a much clearer error signal. However, the reward itself is weak in terms of general dialogue appropriateness. Simply counting repeated tokens does not make a good prediction of general dialogue quality. Repetition can even be appropriate in some cases. While this reward does not improve quality on itself, we experience that low repetition can be learned with reinforcement learning. For PersonaChat it slightly increased from 0.90 to 0.92 over the training, and for KOMODIS it is capped at it's maximum value of 1 throughout the training (including some noise). In both cases, the repetition reward ensures that the generated sequences stick to a non-repetitive behavior, which is, in general, desirable.

Changing the weights of the consistency reward does not have a significant impact on the results. Graphs look very similar for both $f_c = 0.7$ and $f_c = 0.8$ on both datasets. The qualitative result (see Appendix C) also aligns with that. We do not notice any significant improvements for consistency on the generated sequences with raising f_c. This shows that the destructive behavior that we observed when combining rewards, cannot be overcome by weight parametrization. In the following section, we discuss this and the viability of reinforcement learning for these tasks, and propose an interesting experiment to further investigate the issue of combining language modelling rewards in these setups.

6.5. Viability of Reinforcement Learning

Reinforcement learning for dialogue generation has two major drawbacks: The first one is the lack of automated metrics. The strength of reinforcement learning is the possibility to use non-differential loss functions, but without having strong metrics it cannot work well. High fluctuations make the reward signal more noisy, making the difficult task of learning appropriate dialogue behavior even more difficult. We think that reinforcement learning will benefit a lot from better and more precise automated metrics, once they are available in the research community. Second, natural language generation is a very sparse computation process. At each step, the model has to choose one token out of the whole vocabulary, which can contain several thousand elements. That makes the learning process expensive, and reinforcement learning might even need much more data than the current language modelling approaches. This is especially true, when combining rewards, as explained in the previous section.

For the future, we suggest a series of experiments to further investigate the problem of combining different language modelling rewards. Instead of training the difficult task of dialogue generation, it would be interesting to see the performance of language models on simple tasks, for example "creating sentences with or without specific words", or "creating sentences with specific amounts of tokens", and so on. For that, clear metrics are available and do not distort the results. In a second step, these simple tasks need to be combined to see how well the reinforcement learning algorithms learn the combined behavior. The relative increase in capacity that is needed to learn them, should give interesting insights in the nature of reinforcement learning for language generation.

Despite these difficulties, we think that reinforcement learning is a promising algorithm for future language generation, as their strength to make use of any kinds of metrics, is an important step to have control over the key factors for having an appropriate (artificial) conversation.

7. Improving Consistency with Transfer Learning

The experiments and evaluations in this chapter were done by the student Karthik Ajith Kumar in his master thesis "Improving a closed-domain conversational AI using Transfer learning". I supervised the student and was mainly responsible for all ideas and strategies to conduct the experiments. I am responsible for writing all sections of this thesis and the interpretation of the results. The conclusion in this thesis differs from Karthik Kumar's work.

In this chapter, we explore the possibilities of transfer learning between dialogue datasets with a primary focus on consistency towards facts and opinions. Transformer-based models already use transfer learning to enhance their performance: all the models we used in this thesis were pre-trained on unsupervised text data. Fine-tuning these models with specific dialogue datasets is called sequential learning in that context. However, the pre-trained data does not contain any additional context, so that the transfer learning part is only given to the natural language generation itself. We explain these limitations further in Section 7.1. We then discuss our series of experiments for sequential, cumulative and simultaneous learning in Section 7.2 and discuss the results in Section 7.3. For analysing the results we conduct another human evaluation, since automated metrics are not sufficient to determine the consistency accuracy in our experiments.

Smith et al. (2020) showed that combining multiple dialogue datasets in general is beneficial for the performance of neural conversation models. They created a new dataset to combine the different properties of these datasets. We, however, explore transfer learning approaches to see the influence of consistency for models optimized on our KOMODIS dataset.

7.1. Limitations of Data Sources

All available datasets that focus on consistency towards additional context, like Wizard of Wikipedia Dinan et al. (2019c), PersonaChat Zhang et al. (2018) as well as KOMODIS were created with the help of crowdsourcing, which requires a lot of resources for preparation, post processing and the crowdsourcing itself. This limits the sizes of these datasets. However, the training tasks that these datasets offer are challenging and require good generalization from the models. Especially, since the data structure is significantly different from the pre-training data.

Additionally, these datasets have limited variety in dialogue structure in most cases. In the PersonaChat dataset, for example, the dialogues are about to get to know each other. Even if the personality profiles are open domain, that setting itself gives these dataset dialogues specific properties: they are biased towards asking questions about the other person and telling the person about their personality. Our KOMODIS dataset is set in the domain of movies and all dialogues are about discussing a specific movie, and asking and answering questions about it. Forcing these models to reply to content out of these structures, often causes weak performance, or the models try to stick to what they have learned, making the conversation monotonous or even unnatural.

We elaborate the possible advantages of transfer learning regarding these points in the following sections.

7.2. Experiments

All experiments in this chapter are supposed to analyse the performance on KOMODIS. For our baseline model, we use a DialoGPT model (Zhang et al., 2020) fine-tuned on KOMODIS until the loss function converged on validation data, see Section 4.4.4 for more details. The transferring dataset sources we use in this experiments are OpenDialKG (Moon et al., 2019), discussed in Chapter 5.2, EmpatheticDialogues (Rashkin et al., 2019) and Self-Dialogues (Fainberg et al., 2018). They all contain medium-sized non-goal-driven dialogues, and crowdsourced in a specific environment. OpenDialKG dialogues are based on a knowledge graph about movies, books, sports or music and fit well to the idea of KOMODIS. The dialogues from EmpatheticDialogues also have additional context, but in the form of a short description of a situational emotion. Self-Dialogues contains conversations about movies, music, sports and fashion, and the data was collected by tasking the participants to talk with

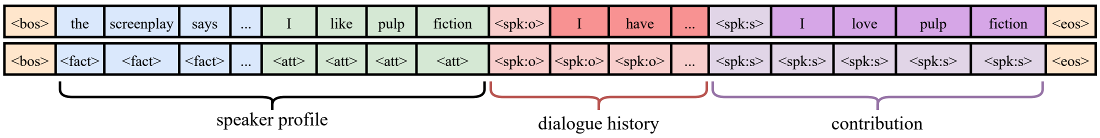

Figure 7.1.: Sequence encoding example for KOMODIS with speaker profile (blue), dialogue history (red) and the contribution (purple) that has to be predicted. Word level (top row) contains the language vocabulary and segment level (bottom row) contains the special tokens to separate the different segments.

themselves (i.e. being in the role of both dialogue partners). There are more suitable datasets available, but we choose to only use three significantly different datasets, as the experiments require a lot of resources and manual human evaluation is required, which limits the amount of experiments further.

We first have to prepare the datasets with a shared encoding strategy, which we elaborate in the following subsection. Then, we conduct three series of experiments, one for each learning strategy, that we discuss in the following subsections.

Common Encoding To make the transfer learning approaches effective, we have to encode the dialogues from the different datasets as similar as possible. Since we want to optimize the model towards the KOMODIS dataset, we choose our encoding from Chapter 4 as the default and developed the others by slightly changing the purpose of the special tokens. Figure 7.1 shows one example encoding for KOMODIS. We separate facts from opinions and differentiate between the two dialogue partners using a segments layer.

OpenDialKG has knowledge graphs as additional input. We reuse the fact segment from our encoding for these graphs, and encode the graph as triples, similar to the encoding from Chapter 5.2. Samples from that dataset don't have an opinion segment. The dialogue history is encoded with no difference. Figure 7.2 shows an example encoding for that dataset at the top. EmpatheticDialogues is the counterpart to OpenDialKG, as it has emotional situations as the context. We encode them in the opinion segment of the encoding. Samples from this dataset don't have a fact segment. Again, dialogue history is encoded with no differences. Finally, the Self-Dialogues are encoded without additional context, as the dataset is context-free. Dialogue history is encoded with no differences. Figure 7.2 shows an example encoding for both datasets (middle and bottom), as well.

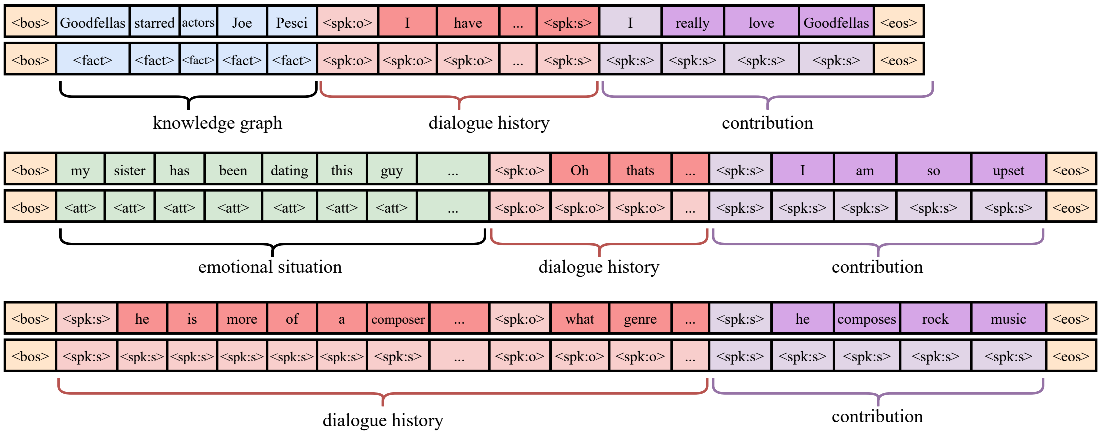

Figure 7.2.: Sequence encoding for the other datasets. Top: OpenDialKG using the fact segment (blue) for the knowledge graphs. Middle: EmpatheticDialogues using the opinion segment (green) for the emotional situations. Bottom: Self-Dialogues without any additional context.

default hyper parameter	
batch size	32
optimizer	Adam
β_1	0.9
β_2	0.999
ε	1×10^{-8}
learning rate	6.25×10^{-6}
dropout	0.1
weight decay	0.01

Table 7.1.: Default hyperparameter settings for all experiments.

Global Settings We split all datasets into 80% training, 10% validation and 10% test data and use perplexity on validation data to track over-fitting. All experiments are done with the DialoGPT model. The model has 117 million parameters with 12 self-attention layer and 768 dimensional embeddings. Hyperparameter were mostly taken from pre-training. Table 7.1 lists the exact parameters. Since we want to primarily evaluate the consistency towards background context, we decided to use the Beam-Search algorithm with 4 beams and simple 3-gram blocking to generate all utterances. Beam-Search is more precise in terms of decoding knowledge, as discussed in Chapter 5.2.

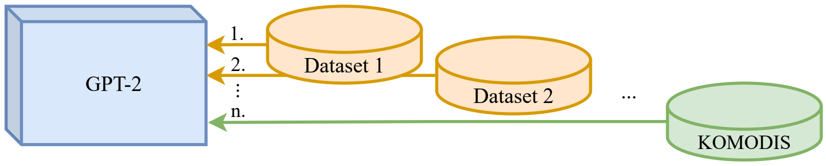

Figure 7.3.: Sequential learning approach with multiple datasets, optimized on KOMODIS. Each dataset is trained separately until the loss converged on the validation set. KOMODIS is used last.

7.2.1. Transfer Learning Strategies

Sequential Learning Sequential learning is probably the most common transfer learning strategy, and the main purpose of it is domain adaption. Here, the model is trained on a non-specific dataset first, and then fine-tuned on the domain-specific afterwards. The most prominent example in natural language processing is BERT (Devlin et al., 2019). It is pre-trained on a very generic, but large dataset and then used with many specific and smaller NLP datasets, to adapt the model to the specific needs.

In our experiments, we follow this idea: We first train a DialoGPT model on the three additional datasets until the loss converges on validation data, and then train the model on KOMODIS as usual. Figure 7.3 visualizes that process. In this series of experiments, we try using only one additional dataset, two additional datasets and all three. For each variant, we repeat the training with different orders. Results are listed and discussed in Section 7.3.2.

For the human evaluation we choose 3 different models: One trained only with 1 additional dataset (EmpatheticDialogues), one with 2 additional datasets (EmpatheticDialogues and Self-Dialogues) and one with all three. We refer to them as *sequential-1*, *sequential-2* and *sequential-2*, respectively.We experimented with all permutations, but had to reduce the number of models for the human evaluation. Therefore, we chose 3 representative models, as the order of the datasets seem to have less influence on the results.

Cumulative Learning With cumulative learning, all datasets are combined to one bigger set. It was used in the approach from Smith et al. (2020), but instead of merging datasets equally with additional label-effort, we homogenise the encoding strategies across the datasets and just shuffle samples together. We also only validate with KOMODIS samples, as our goal is to optimize the model for that dataset. Figure 7.4 visualizes that process. Results are listed and discussed in Section 7.3.2. We refer to that model as *cumulative* in the evaluation.

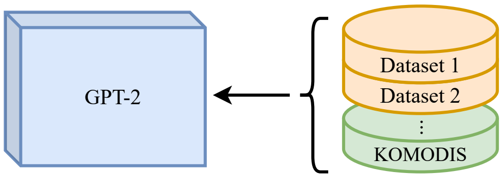

Figure 7.4.: Cumulative learning approach with multiple datasets, optimized on KOMODIS. All datasets are combined to one final dataset. Samples are shuffled and the model is trained until the loss converged for KOMODIS on the validation data.

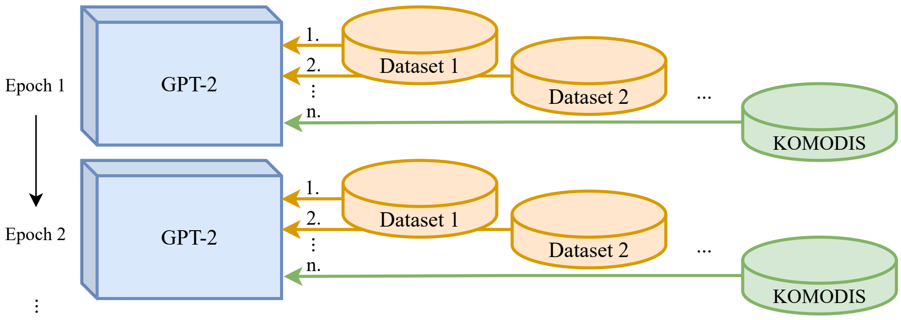

Figure 7.5.: Simultaneous learning approach with multiple datasets, optimized on KOMODIS. Each dataset is used once per epoch, with KOMODIS being the last. The model is trained until the loss on the KOMODIS dataset converged on validation data.

Simultaneous Learning Simultaneous learning is a mixture of the previous ones. Each dataset is handled separately, but for each epoch, a new dataset is chosen. In that process, KOMODIS is taken last, and we stop the training after the validation loss on KOMODIS data converged. Figure 7.5 visualizes the process. The order does not change after each full epoch. Results are listed and discussed in Section 7.3.2. We refer to that model as *simultaneous* in the evaluation.

7.2.2. Naturalness and Context Consistency

For all transfer learning strategies we compare the consistency towards context from KOMODIS (separated into facts and opinions), as well as the general naturalness of the dialogues. For that, we use the same approach in the human evaluation, as in Chapter 5.2: We count fact and opinion occurrences, and ask the participants to label them whether they are correct or incorrect towards the context. Additionally, we generically ask for naturalness, consistency and how interesting the dialogue partner appear in the conversations. Finally, we force the participants to choose the more engaging dialogue partners, given pairs of dialogues, which we call *win ratio* in the evaluation.

7.2.3. Hallucination

Generating factual or personal information that is not specified in the background context is called *hallucination*. Usually, this is an unwanted behaviour, because the hallucinated facts are uncontrollable and often wrong. Hallucinating personal information is also uncontrollable, but not wrong by default, because personal information is not globally known. Therefore, a system that contributes with made-up personal information (“I like him alot!”) might be perceived more positive, than not having an opinion at all (“I don’t know.”).

We want to see the influence of transfer learning on hallucination. Therefore, we ask the participants to count and cluster hallucinations in our human evaluation. Having many dialogues (from other datasets) that not specifically make use of external information, might hurt the training process on the KOMODIS data. We list the results in Table 7.3 and discuss them in Subsection 7.3.2.

7.3. Results

As discussed in Chapter 3, automated evaluation of consistency towards background context lack adequate automatic metrics. Therefore, we only report perplexity on test data for our models and conduct another human evaluation. We present the setting and results of it in Subsection 7.3.1. Thereafter, we evaluate the different transfer learning strategies based on the results in Subsection 7.3.2. Finally, we conclude with a statement on the viability of transfer learning in our experiment setting.

7.3.1. Human Evaluation

The human evaluation for this series of experiments was done similar to our evaluation from the knowledge graph encoding experiments, described in Section 5.6.1. This evaluation study was also managed by researchers not involved in setting up the models and experiments.

Method 20 participants not familiar with our research and the goals of the study took part in the human evaluation. 4 Participants were women, and 16 participants stated that they already have experience with various forms of chatbots. 6 participants were 18–35 years old, 10 were 36–50 years old, and 4 participants were older than 50 years. The procedure is similar to the previous human evaluation, but we added for additional questions that have to be answered by the participants:
(1) "Is there a hallucination in that utterance?" Possible answers are *yes* and *no*. A hallucination was explained as a specific fact or opinion, that is not given in the background context.
The following 3 are follow-up questions, if the the previous one was answered with *yes*:
(2) "Is the hallucination about the movie or another domain?" Possible answers are *movie* or *other*. We want to differentiate between content in and out of the KOMODIS domain.
(3) "Is the hallucination about a fact or an opinion?" If it's an in-domain hallucination, we want to further differentiate between facts and opinions, according to the design of the KOMODIS dataset.
(4) "Is the hallucination referring to a given entity or a full new creation?" Here, we want to differentiate whether the hallucination is related to the entities from the given background context, or not.
The questionnaire contains a survey guide and a set of dialogue pairs to evaluate, similar to our previous human evaluation. Details can be found in Section 5.6.1.

Annotator Results Table 7.2 lists the results for context consistency and naturalness from the human evaluation, and 7.3 lists the hallucination results. For the baselines, we added samples from the original dataset, that we refer to as *dataset*. That serves as an upper threshold, as the model should not become better as the observed dialogues. And we added a model trained without transfer learning, that we refer to as *no-transfer* in the following evaluation.

7.3.2. Transfer Learning Strategies

The dataset experiment has highest win-ratio and least amount of incorrect knowledge, as expected. They also got highest values for naturalness, consistency and interestingness. However, it has highest amount of incorrect opinions, which is very unexpected. It also has the lowest amount of correct opinions. We found that participants did not only evaluate the opinions given by the background context, but also any kinds of opinions that can occur in a conversation. That interpretation issue did not occur in our large human evaluation from

experiment	**ppl**	**win ratio (%)**	**precision / incorrect (%) knowledge**	**opinions**	**natural**	**agreements consistent**	**interesting**
dataset	-	**68.4**	**73.4 / 6.3**	70.6 / 20.0	**5.1**	**5.1**	**5.0**
no-transfer	7.21	41.7	60.9 / 21.8	**86.4 / 8.0**	4.9	4.5	4.4
sequential-1	7.22	34.3	58.9 / 24.4	**89.5** / 5.8	4.4	4.5	4.2
sequential-2	7.16	42.2	62.2 / 23.3	87.0 / **3.9**	**5.1**	4.8	4.6
sequential-3	7.14	53.7	**73.6 / 22.0**	85.4 / 11.0	**5.1**	**4.9**	**4.7**
simultaneous	7.32	**62.8**	54.7 / 27.4	85.9 / 6.5	4.6	4.4	4.3
cumulative	**7.12**	61.5	60.8 / 25.5	80.5 / 9.2	5.0	4.9	4.5

Table 7.2.: Perplexity on the test set (lower is better) and human evaluation results for models trained on KOMODIS. Metrics explained in Subsection 5.6.1. Agreements are on a 7-point Likert scale (higher is better).

Chapter 5.2. The human evaluation manager might have communicated that specific part to the participants insufficiently. Since dataset dialogues usually have more variety, there is more opinionated content in it that was labeled as *wrong* from the participants, as they could not find it in the background context. Therefore, we exclude the opinion results from our following evaluation discussion.

Sequential-3 has the highest consistency towards facts and best agreement values. The last step of this transfer learning algorithm is the fine-tuning on the target dataset KOMODIS. The advantage of this approach is the same as with all Transformer models: The model weights are optimized towards KOMODIS without disturbing influences of other samples.

To evaluate the influence of transfer learning we have to compare these models with our model baseline *no-transfer*. In our experiments, sequential learning can only exceed the baseline when using three additional datasets. Win-ratio, knowledge precision and the agreements increase with the number of additional datasets. *sequential-1* got the weakest results with only 34.3% win-rate, even lower as the baseline with 41.7%. *sequential-3* has highest knowledge precision across all experiments and also produced the least amount of wrong facts. It seems that highest fact consistency (on dialogues in a similar setting as KOMODIS) can be achieved by focussing on the KOMODIS training data. It has significantly higher fact consistency than the *no-transfer* baseline, and an increasing number of additional datasets has a positive influence on consistency. With that approach, the model is able to transfer knowledge. However, only using one additional dataset causes a performance drop on all metrics, compared to *no-transfer*. We think that only using one additional dataset overfits the model, reducing the pre-training benefit from the original Transformer model, due to catastrophic forgetting. More additional datasets, however, seem to generalize good enough

experiment	occurrences	in-domain (movie)	fact-based	context-related (entity)
dataset	22.4%	57.4%	45.6%	39.7%
no-transfer	18.4%	56.6%	54.7%	39.6%
sequential-1	19.4%	75.0%	62.5%	35.7%
sequential-2	25.9%	64.9%	56.8%	43.2%
sequential-3	28.4%	57.0%	45.2%	35.5%
simultaneous	27.0%	62.4%	45.2%	36.6%
cumulative	25.9%	51.8%	49.4%	40.0%

Table 7.3.: Hallucination results from human evaluation. Each column refers to one of the 4 additional questions stated in Section 7.3.1 in the method paragraph. Lower values does not necessarily mean higher quality. We discuss this in Section 7.3.2.

on dialogues, to increase the performance on the target dialogue dataset.

Simultaneous has highest win-rate with 62.8% but did not perform well in the other tasks: It only produced 54.7% correct facts and the highest amount of wrong facts (27.4%). The most frequent reasons why participants chose that model, were: “Asked counter questions” and “coherent conversation structure”. Shuffling the datasets per epoch does not specifically optimize the model weights towards KOMODIS. Therefore, in order to perform good on KOMODIS data, successful transfer learning is important. The significant drop in fact consistency aligns well with our assumption. However, according to the win-ratio, the overall performance from this model is very good. We conclude that the simultaneous approach is able to transfer knowledge in the form of generic dialogue behaviour, but does not help on fact consistency.

The *cumulative* approach seems appealing as well, with a win-rate of 61.5%. It has very similar knowledge precision to the $no-transfer$ baseline. Here, the data shuffling is done on batch level. Results are similar to the *simultaneous* model. We assume that the reasons are similar as well. The cumulative approach is able to transfer generic dialogue behaviour, but since it is not optimized on KOMODIS in the final step, it is difficult to achieve high fact consistency.

7.3.3. Hallucination

One goal of the human evaluation is to analyse the impact of transfer learning on hallucination. Table 7.3 lists the results from the human evaluation. Participants detected a hallucination in every fifth utterance (22.4%) from the *dataset* experiment. The *no-transfer* model has slightly reduced hallucinations with a value of 18.4%. Generated dialogues are always less diverse (and less interesting) as the human-human dialogues and the reduced hallucination might be caused by the less surprising utterances. General hallucination increases with transfer-learning as expected. For sequential learning the amount of additional data seems correlate with hallucination (*sequential-3* has a rate of 28.4%) and simultaneous and cumulative learning also produce similar values (27.0% and 25.9%, respectively).

However, the hallucination types do not change significantly, measured by the other metrics that are listed in Table 7.3. The *no-transfer* baseline and the *dataset* have around 57% in-domain hallucinations, meaning that they produce slightly more movie-related hallucinations than out-of-domain statements. The *sequential-1* and *sequential-2* models produce up to 75% in-domain hallucinations, but the other ones show a behaviour that is very similar to the baselines. The same holds for the ratio between facts and opinions: All experiments produce equally distributed hallucinations within an average difference of 5%. Furthermore, all models produce context-related hallucinations in around 40% of the cases.

Transfer learning increases hallucination in general, but that does not seem to have a significant impact on the quality of these models. When comparing the results from Table 7.2 with the hallucination ratio, there is no significant correlation between win-ratio or any of the three agreement metrics, except for the three sequential models, but we assume that this correlation comes from the overall quality of them and not from hallucination.

7.3.4. Viability of Transfer Learning

Our experiments show that transfer learning using additional dialogue datasets is beneficial. To preserve very specific dialogue characteristics, like the fact consistency of KOMODIS, sequential learning seems to perform best. It is already an established and powerful pre-training step, like in the Transformer models. Using sequential learning with additional dialogue datasets having similar sizes as the target dataset, it is important to provide a sufficient amount of *other* data, to make sure that the model does not overfit towards the new data in the process. According to our experiments, at least to additional datasets are required to match the performance of the target dataset alone, and three additional datasets to achieve significant improvements with sequential learning. However, these values shouldn't be generalized, as other datasets might have different interferences.

Using transfer learning algorithms that do not prioritize the target dataset (like simultaneous or cumulative approaches) are beneficial as well, but in a more generic way: According to our experiments, the model is not able to utilize the additional knowledge to improve fact consistency (one of the core characteristics of KOMODIS), but participants in the human evaluation rated these dialogues high in terms of general positive feeling (win-ratio). These results are consistent with the experiments from Smith et al. (2020). They reported a benefit, combining datasets to create an improved dialogue experience. Fact consistency is only one part of being an appropriate dialogue partner, and an ultimate (not yet existing) neural conversation model needs to learn all aspects of it.

8. Conclusion

In this thesis, we worked out that neural conversation models should be fluent, informative, consistent towards background context, coherent and follow a social norm, derived from the *Coorperative Principle* by Grice (1975). We further investigated consistency towards background context by introducing a new context augmented conversational dataset KOMODIS, and explored encoding strategies that utilize the idea of graph neural networks, to feed the context into models efficiently. In our final experiments, we analyzed the effectiveness of reinforcement and transfer learning, with a focus on consistency. The conclusions drawn from this work can be split into three main aspects: The current state of neural conversation models, the role of consistency to improve conversation appropriateness and the potential of reinforcement learning for these models.

Current state of neural conversation models Neural conversation models received increased attention over the last few years. Research in this area is twofold: One approach focusses on improving performance by using more and more available data and computing resources. Recently, Shuster et al. (2022) introduced the SeeKeR [1] model.

This model produces very natural contributions and can keep conversations ongoing over multiple turns. Unfortunately, we cannot compare the performance between models from different studies: The metrics provided by the authors rely on human annotations, because the complexity of appropriate contributions cannot be measured well by known automated metrics. And since every human evaluation setup differs, they are not comparable in most cases. SeeKeR is a GPT-2 model (Radford et al., 2019) further pre-trained on 1.5 billion training examples from Reddit [2] and an additional 100 billion tokens from various English-language corpora similar to Liu et al. (2019). The authors then used many other dialogue datasets for fine-tuning, using 64 GPUs with 32GB memory each. To achieve knowledgeable conversations, they used the internet search query generation that we explained in Section 3.4.3. The authors are able to use numerous resources in the form of data and hardware

[1] `https://github.com/facebookresearch/ParlAI/tree/main/projects/seeker`
[2] `https://files.pushshift.io/reddit/`

to achieve these results. This proves the complexity of the task of contributing appropriately to a conversation. The generalization effort that a neural network has to achieve is enormous.

The other research area focuses on solving specific problems that can be categorized into fluency, informativity, consistency, coherence and following the social norm. Usually, these models are trained with significantly less resources and achieve promising results in their specific area. The ultimate goal, however, that we motivated in our introduction to build a model that can have *appropriate* and therefore human-like conversations with real persons requires models to master each of these categories. Our operationalization of the Conversational Maxims from Grice (1975) as Demands on Conversation Models in Chapter 3 helped to sort all these approaches, and showed that especially consistency, coherence and following a social norm need further attention in terms of data, new approaches and reliable evaluation methods.

Overall, we think both approaches are valuable: Models such as SeeKeR show the effectiveness of big data sources, but research towards specific dialogue aspects allows a more profound understanding of what is needed to achieve that goal.

Consistency towards background context In Chapters 4 and 5 we showed that the most promising way of feeding knowledge into a neural conversation model is to provide context and specifically train models to make use of it. Neural networks are not designed to store knowledge in their weights and providing it in a human-readable format makes it easy to adapt or augment knowledge later, without the need of re-training the models. Learning consistency with respect to background context has two main challenges: It requires a precise pre-estimation step, as knowledge bases can be huge. It is impossible to feed the Wikipedia database into a neural model, for example, or the whole IMDb in our case. Secondly, training data is limited, as the task of using background context in a consistent way requires specifically labeled data. We contributed to this by providing a new dataset that combines factual and personal information, and proposed a new concise encoding based on graph neural networks to increase the amount of data that can be handled by a Transformer-based neural conversation model.

Consistency with respect to background context plays an important role in a dialogue system. It makes a conversation more informative and therefore engaging, as it allows the system to share knowledge with the user. It also enables the system to have a consistent opinion, and allows controlling the personality of the system, by changing the opinions in the context.

Since our data is domain-specific (movies) and medium-sized (around 11k dialogues), it only enables the conversation system to strictly talk about movies in a slightly biased way, according to the dialogues. Therefore, we experimented with transfer learning and conclude that combining our approach with more divers datasets to build a more robust system, and thereby combining the two approaches mentioned above, is promising.

Tackling consistency is an interdisciplinary challenge that also includes efficient and precise knowledge retrieval algorithms. Our concise graph encoding opens up broader retrieval approaches, as it allows the use of more context in the generation process. But our results also show that pre-training is essential. Transformer-based models have large training capacities and can overfit fast. For further research, we suggest finding ways to pre-train models with our encoding in an unsupervised way, as we expect to see quality improvements with a proper pre-training.

The lack of reliable automated metrics, especially for consistency with respect to background knowledge, is a significant problem. Almost all contributions in that field have to conduct some form of human evaluation, and in order to compare approaches, the authors have to add these approaches to their evaluation, which is often impractical. We proposed an NLI model for consistency specifically for KOMODIS, but even these models, that are tailored towards a specific task, do not perform good enough to make reliable comparisons. Finding ways to automatically evaluate consistency would probably also generate knowledge for new training strategies, since most training algorithms still rely on the standard language modelling loss. Error signals based on a token-by-token prediction on the ground truth might not be enough to learn the whole complexity of conversation. They also do not support the fact that every dialogue can be continued in many different ways appropriately, nor the problem of error accumulation at decoding without a ground truth.

Potential of reinforcement learning Reinforcement learning has the potential to overcome the problems of having non-optimal error signals for training. In Chapter 6 we investigated current approaches for neural generation models and conducted experiments on our own data. While reinforcement learning is promising, it still has two major drawbacks: The first one is the previously discussed lack of automated metrics. The strength of reinforcement learning is the possibility to use non-differential loss functions, but without having strong metrics it cannot work well. Second, natural language generation is a very sparse computation process. At each step, the model has to choose one token out of the whole vocabulary, which can contain several thousand elements. That makes the learning process expensive, and reinforcement learning might even need much more data than the current language modelling approaches. Another issue that we observed by training conversation models with reinforcement learning is the problems that arise when combining different rewards. It appears that they interfere with each other, as combining rewards significantly reduces their learning process. While Mesgar et al. (2021) suggest that this effect naturally occurs, because learning only one reward at a time causes overfitting, we argue that the negative interference could come from a weak reward signal, that cannot account for the complexity of what appropriateness is in a conversation. We suggest conducting more experiments on that behavior to find solutions, as working reinforcement learning algorithms would allow to explicitly train models to be more consistent towards background context (and towards other demands as well), instead of only learning token probabilities by observing ground truth contributions.

List of Abbreviations

AMT Amazon Mechanical Turk
HIT Human Intelligence Task
IMDb Internet Movie Database
KOMODIS Knowledgable and Opinionated MOvie DIScussions
POS Parts of Speech
RNN Recurrent Neural Network
NLI Natural Language Inference
BERT Bidirectional Encoder Representations from Transformers
GPT Generative Pre-trained Transformer
LSTM Long Short-Term Memory
KNN K-Nearest Neighbours
InI Inconsistency Identification
BLEU BiLingual Evaluation Understudy
CVAE Conditional Variational Auto Encoder
KL Kullback Leibler
RUBER Referenced metric and Unreferenced metric Blended Evaluation Routine
GRADE Graph-enhanced Representations for Automatic Dialogue Evaluation
HMI Human-Machine Interfaces
ASR Automatic Speech Recognition
NLU Natural Language Understanding
NLG Natural Language Generation
TTS Text to Speech
NLP Natural Language Processing

Bibliography

Ameixa, D., Coheur, L., Fialho, P., and Quaresma, P. (2014). Luke, i am your father: Dealing with out-of-domain requests by using movies subtitles. In *IVA*.

Banchs, R. E. (2012). Movie-DiC: a movie dialogue corpus for research and development. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 203–207, Jeju Island, Korea. Association for Computational Linguistics.

Bao, S., He, H., Wang, F., Wu, H., and Wang, H. (2020). PLATO: Pre-trained dialogue generation model with discrete latent variable. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 85–96, Online. Association for Computational Linguistics.

Bast, H., Bäurle, F., Buchhold, B., and Haußmann, E. (2014). Easy access to the freebase dataset. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW 14 Companion, page 95–98, New York, NY, USA. Association for Computing Machinery.

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Bollacker, K., Cook, R., and Tufts, P. (2007). Freebase: A shared database of structured general human knowledge. *Proceedings of the national conference on Artificial Intelligence*, 22(2):1962.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

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. *CoRR*, abs/2005.14165.

Chen, X., Chen, F., Meng, F., Li, P., and Zhou, J. (2021). Unsupervised knowledge selection for dialogue generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1230–1244, Online. Association for Computational Linguistics.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Dai, S., Gan, Z., Cheng, Y., Tao, C., Carin, L., and Liu, J. (2021). APo-VAE: Text generation in hyperbolic space. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 416–431, Online. Association for Computational Linguistics.

Deriu, J., Rodrigo, A., Otegi, A., Echegoyen, G., Rosset, S., Agirre, E., and Cieliebak, M. (2021). Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, 54(1):755–810.

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

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *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. Association for Computational Linguistics.

Dinan, E., Abercrombie, G., Bergman, A. S., Spruit, S., Hovy, D., Boureau, Y.-L., and Rieser, V. (2021). Anticipating safety issues in e2e conversational ai: Framework and tooling.

Dinan, E., Humeau, S., Chintagunta, B., and Weston, J. (2019a). Build it break it fix it for dialogue safety: Robustness from adversarial human attack. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4537–4546, Hong Kong, China. Association for Computational Linguistics.

Dinan, E., Logacheva, V., Malykh, V., Miller, A. H., Shuster, K., Urbanek, J., Kiela, D., Szlam, A., Serban, I., Lowe, R., Prabhumoye, S., Black, A. W., Rudnicky, A. I., Williams, J., Pineau, J., Burtsev, M. S., and Weston, J. (2019b). The second conversational intelligence challenge (convai2). *CoRR*, abs/1902.00098.

Dinan, E., Roller, S., Shuster, K., Fan, A., Auli, M., and Weston, J. (2019c). Wizard of wikipedia: Knowledge-powered conversational agents. *ArXiv*, abs/1811.01241.

Fainberg, J., Krause, B., Dobre, M. S., Damonte, M., Kahembwe, E., Duma, D., Webber, B. L., and Fancellu, F. (2018). Talking to myself: self-dialogues as data for conversational agents. *ArXiv*, abs/1809.06641.

Fan, A., Gardent, C., Braud, C., and Bordes, A. (2021). Augmenting transformers with KNN-based composite memory for dialog. *Transactions of the Association for Computational Linguistics*, 9:82–99.

Galetzka, F., Eneh, C. U., and Schlangen, D. (2020). A corpus of controlled opinionated and knowledgeable movie discussions for training neural conversation models. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 565–573, Marseille, France. European Language Resources Association.

Galetzka, F., Rose, J., Schlangen, D., and Lehmann, J. (2021). Space efficient context encoding for non-task-oriented dialogue generation with graph attention transformer. In *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 7028–7041, Online. Association for Computational Linguistics.

Gao, X., Lee, S., Zhang, Y., Brockett, C., Galley, M., Gao, J., and Dolan, B. (2019). Jointly optimizing diversity and relevance in neural response generation. In *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 1229–1238, Minneapolis, Minnesota. Association for Computational Linguistics.

Ghazarian, S., Wei, J., Galstyan, A., and Peng, N. (2019). Better automatic evaluation of open-domain dialogue systems with contextualized embeddings. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 82–89, Minneapolis, Minnesota. Association for Computational Linguistics.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. *34th International Conference on Machine Learning, ICML 2017*, 3:2053–2070.

Godfrey, J. J. and Holliman, E. (1997). Switchboard-1 Release 2.

Grice, H. P. (1975). Logic and conversation. In Cole, P. and Morgan, J. L., editors, *Syntax and Semantics: Vol. 3: Speech Acts*, pages 41 – 58. Brill, Leiden, The Netherlands.

Gu, X., Cho, K., Ha, J.-W., and Kim, S. (2019). DialogWAE: Multimodal response generation with conditional wasserstein auto-encoder. In *International Conference on Learning Representations*.

Heeman, P. A. and Hirst, G. (1995). Collaborating on referring expressions. *Computational Linguistics*, 21(3):351–382.

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

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration.

Honnibal, M. and Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Huang, L., Ye, Z., Qin, J., Lin, L., and Liang, X. (2020). GRADE: Automatic Graph-Enhanced Coherence Metric for Evaluating Open-Domain Dialogue Systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9230–9240, Online. Association for Computational Linguistics.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y., Madotto, A., and Fung, P. (2022). Survey of hallucination in natural language generation.

Kassner, N., Tafjord, O., Schütze, H., and Clark, P. (2021). BeliefBank: Adding memory to a pre-trained language model for a systematic notion of belief. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8849–8861, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Komeili, M., Shuster, K., and Weston, J. (2021). Internet-augmented dialogue generation. *CoRR*, abs/2107.07566.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2016a). A diversity-promoting objective function for neural conversation models. In *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. Association for Computational Linguistics.

Li, J., Galley, M., Brockett, C., Spithourakis, G., Gao, J., and Dolan, B. (2016b). 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, M., Roller, S., Kulikov, I., Welleck, S., Boureau, Y.-L., Cho, K., and Weston, J. (2020). Don't say that! making inconsistent dialogue unlikely with unlikelihood training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4715–4728, Online. Association for Computational Linguistics.

Lin, X., Jian, W., He, J., Wang, T., and Chu, W. (2020). Generating informative conversational response using recurrent knowledge-interaction and knowledge-copy. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 41–52, Online. Association for Computational Linguistics.

Liu, A., Sap, M., Lu, X., Swayamdipta, S., Bhagavatula, C., Smith, N. A., and Choi, Y. (2021). DExperts: Decoding-time controlled text generation with experts and anti-experts. In *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 6691–6706, Online. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

Lowe, R., Noseworthy, M., Serban, I. V., Angelard-Gontier, N., Bengio, Y., and Pineau, J. (2017). Towards an automatic Turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126, Vancouver, Canada. Association for Computational Linguistics.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Mehri, S., Razumovskaia, E., Zhao, T., and Eskenazi, M. (2019). Pretraining methods for dialog context representation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3836–3845, Florence, Italy. Association for Computational Linguistics.

Meister, C., Cotterell, R., and Vieira, T. (2020). If beam search is the answer, what was the question? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2173–2185, Online. Association for Computational Linguistics.

Mesgar, M., Simpson, E., and Gurevych, I. (2021). Improving factual consistency between a response and persona facts. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 549–562, Online. Association for Computational Linguistics.

Moon, S., Shah, P., Kumar, A., and Subba, R. (2019). OpenDialKG: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 845–854, Florence, Italy. Association for Computational Linguistics.

Niwattanakul, S., Singthongchai, J., Naenudorn, E., and Wanapu, S. (2013). Using of jaccard coefficient for keywords similarity. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, pages 380–384.

Paulus, R., Xiong, C., and Socher, R. (2018). A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf.*

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

Rashkin, H., Smith, E. M., Li, M., and Boureau, Y.-L. (2019). Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381, Florence, Italy. Association for Computational Linguistics.

Ritter, A., Cherry, C., and Dolan, W. B. (2011). Data-driven response generation in social media. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 583–593.

Roller, S., Dinan, E., Goyal, N., Ju, D., Williamson, M., Liu, Y., Xu, J., Ott, M., Smith, E. M., Boureau, Y.-L., and Weston, J. (2021). Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, Online. Association for Computational Linguistics.

Sankar, C., Subramanian, S., Pal, C., Chandar, S., and Bengio, Y. (2019). Do Neural Dialog Systems Use the Conversation History Effectively? An Empirical Study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 32–37, Florence, Italy. Association for Computational Linguistics.

Schick, T., Udupa, S., and Schütze, H. (2021). Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp. *Computing Research Repository*, arXiv:2103.00453.

Schlangen, D., Diekmann, T., Ilinykh, N., and Zarrieß, S. (2018). slurk–a lightweight interaction server for dialogue experiments and data collection. In *Short Paper Proceedings of the 22nd Workshop on the Semantics and Pragmatics of Dialogue (AixDial/semdial 2018)*.

Searle, J. R. (1969). *Speech Acts*. Cambridge University Press.

See, A., Roller, S., Kiela, D., and Weston, J. (2019). What makes a good conversation? how controllable attributes affect human judgments. In *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 1702–1723, Minneapolis, Minnesota. Association for Computational Linguistics.

Serban, I., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Shuster, K., Komeili, M., Adolphs, L., Roller, S., Szlam, A., and Weston, J. (2022). Language models that seek for knowledge: Modular search and generation for dialogue and prompt completion.

Smith, E. M., Williamson, M., Shuster, K., Weston, J., and Boureau, Y.-L. (2020). Can you put it all together: Evaluating conversational agents' ability to blend skills. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2021–2030, Online. Association for Computational Linguistics.

Solaiman, I. and Dennison, C. (2021). Process for adapting language models to society (PALMS) with values-targeted datasets. *CoRR*, abs/2106.10328.

Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.-Y., Gao, J., and Dolan, B. (2015). A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205, Denver, Colorado. Association for Computational Linguistics.

Tao, C., Mou, L., Zhao, D., and Yan, R. (2018). RUBER: An Unsupervised Method for Automatic Evaluation of Open-Domain Dialog Systems. In McIlraith, S. A. and Weinberger, K. Q., 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.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vinyals, O. and Le, Q. (2015). A neural conversational model. *ICML Deep Learning Workshop, 2015*.

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

Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., and Weston, J. (2020). Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

Williams, A., Nangia, N., and Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *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. Association for Computational Linguistics.

Williams, J., Raux, A., and Henderson, M. (2016). The dialog state tracking challenge series: A review. *Dialogue and Discourse*.

Wittkenstein, L. (1953). *Philosophische Untersuchungen*.

Wolf, T., Sanh, V., Chaumond, J., and Delangue, C. (2019). Transfertransfo: A transfer learning approach for neural network based conversational agents.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Xiao, Y. and Wang, W. Y. (2021). On hallucination and predictive uncertainty in conditional language generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2734–2744, Online. Association for Computational Linguistics.

Xu, J., Ju, D., Li, M., Boureau, Y., Weston, J., and Dinan, E. (2020). Recipes for safety in open-domain chatbots. *CoRR*, abs/2010.07079.

Xu, X., Dušek, O., Konstas, I., and Rieser, V. (2018). Better Conversations by Modeling, Filtering, and Optimizing for Coherence and Diversity. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3981–3991, Brussels, Belgium. Association for Computational Linguistics.

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?

Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., Gao, J., Liu, J., and Dolan, B. (2020). DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

Zhao, T., Zhao, R., and Eskenazi, M. (2017). Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664, Vancouver, Canada. Association for Computational Linguistics.

Zhao, X., Wu, W., Xu, C., Tao, C., Zhao, D., and Yan, R. (2020). Knowledge-grounded dialogue generation with pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3377–3390, Online. Association for Computational Linguistics.

Zhou, L., Gao, J., Li, D., and Shum, H.-Y. (2020). The design and implementation of XiaoIce, an empathetic social chatbot. *Computational Linguistics*, 46(1):53–93.

Zhou, W., Li, Q., and Li, C. (2021). Learning from perturbations: Diverse and informative dialogue generation with inverse adversarial training. In *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 694–703, Online. Association for Computational Linguistics.

Zhu, Y., Kiros, R., Zemel, R., 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. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, Los Alamitos, CA, USA. IEEE Computer Society.

A. AMT Introduction Slides

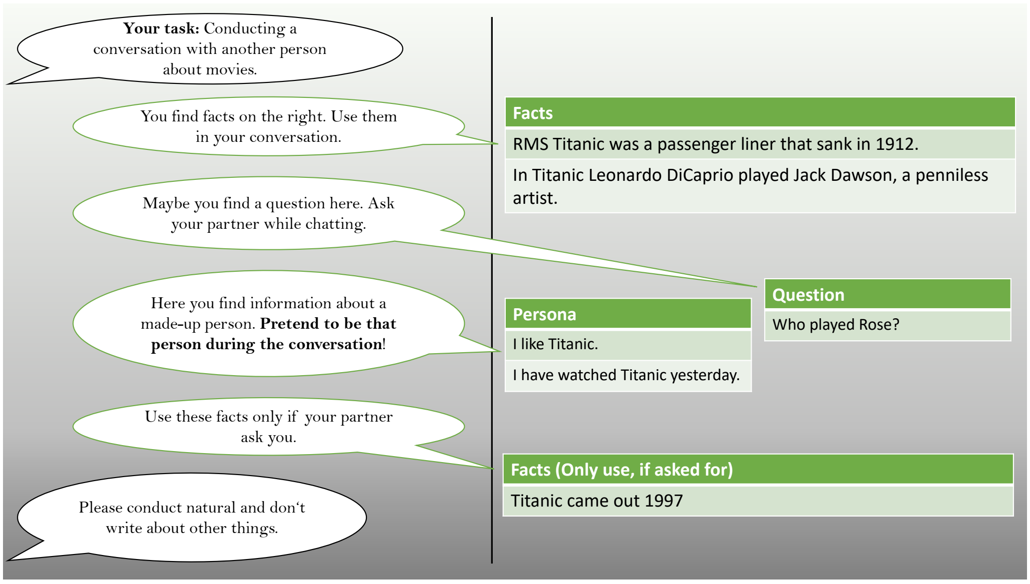

Figure A.1.: First introduction slide for our data collection setup with an overview of the task and an explanation of the speaker profiles.

Figures A.1, A.2, A.3, A.4 and A.5 show our instruction slides for collecting the KOMODIS dataset. Participants from AMT are tasked to go through each slide before accepting the HIT. We first give an overview of the whole task, and then explain the payment on the second slide. Since the 'super bonus' is not easy to understand, we explain it the additional third slide. After that, we summed up some more hints on slide 4, which we found to be useful during our first iterations on the collection setup. The final slide gives technical hints and the necessary information about their data being logged.

Figures A.6 and A.7 show the instruction slides for collecting the entailment statements as an extension for KOMODIS. Again, the participants are tasked to go through each slide before accepting the HIT. The first slide explains the task, and the second slide presents some very important examples, as the task itself is not easy to understand, as we found out in first iterations.

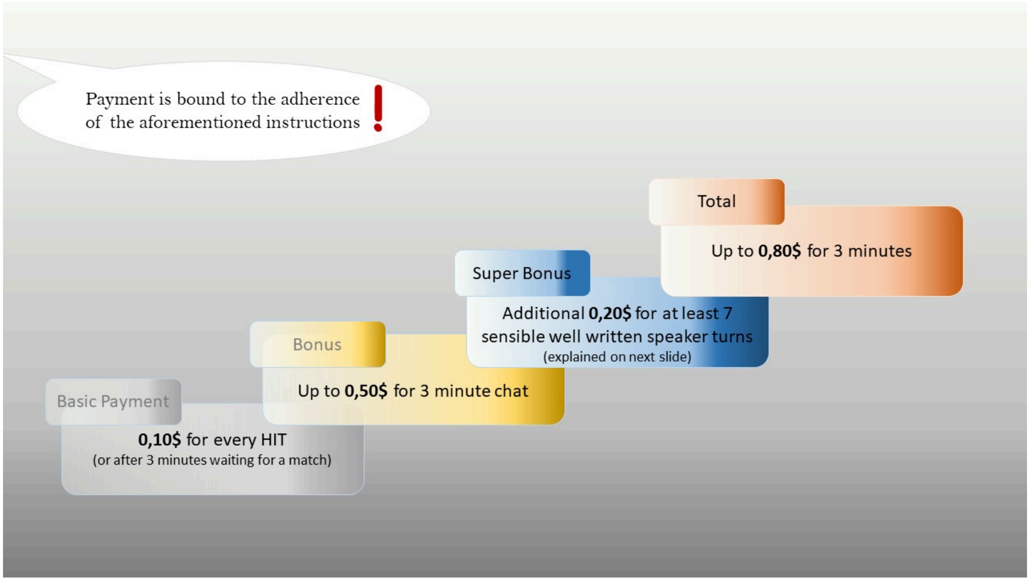

Figure A.2.: Second introduction slide for our data collection setup. It explains the payment for the participants. Differentiation of bonus and super bonus aims to engage the participants to create appropriate dialogues.

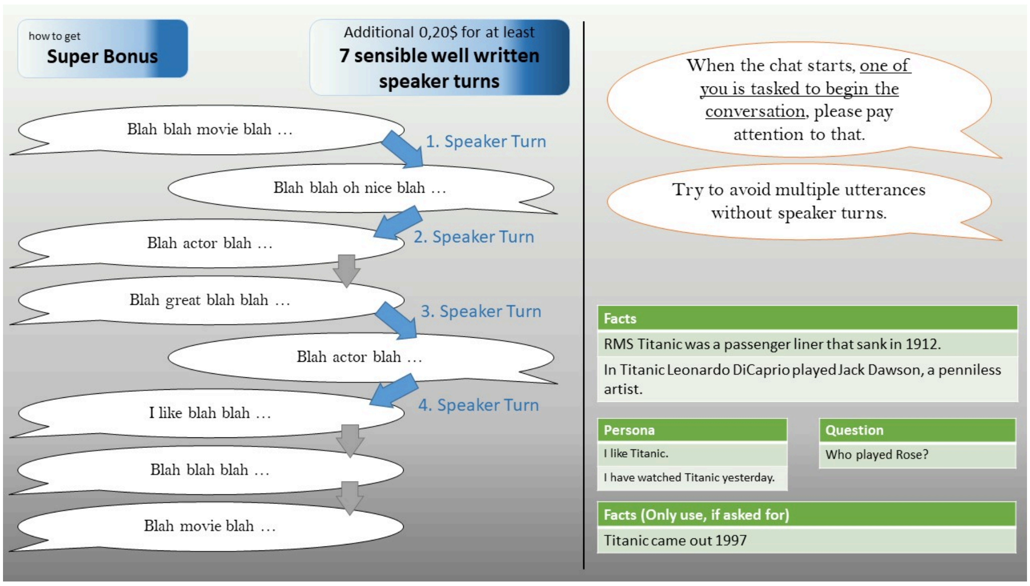

Figure A.3.: Third introduction slide for our data collection setup. It explains what we call the super bonus. The term speaker turn is not common, so we visualize it on the left.

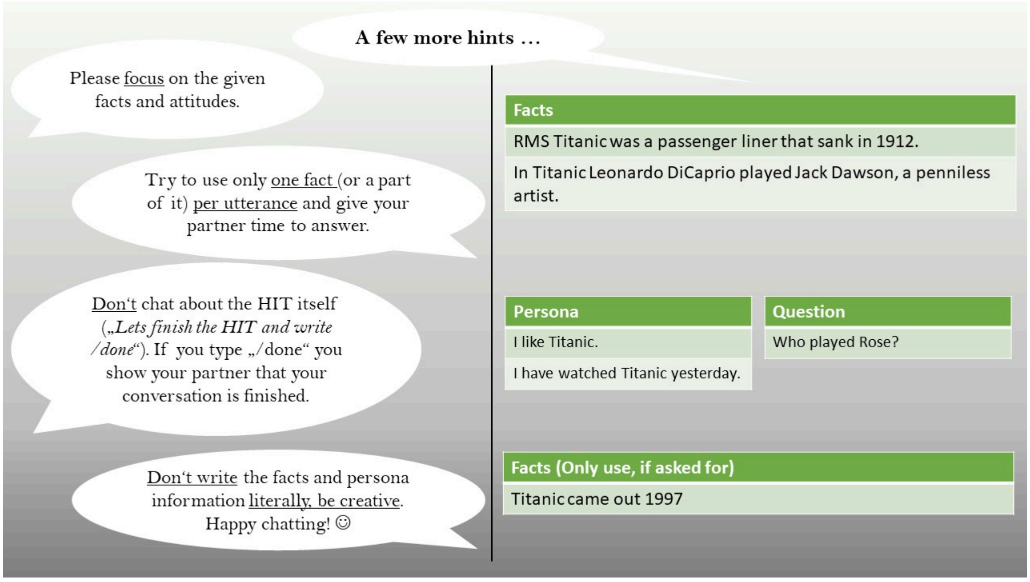

Figure A.4.: Fourth introduction slide for our data collection setup. The purpose of this slide is to help the participants to understand what the goal of their work will be. We added a few hints for them in a short and simple way, to increase the chances to be read.

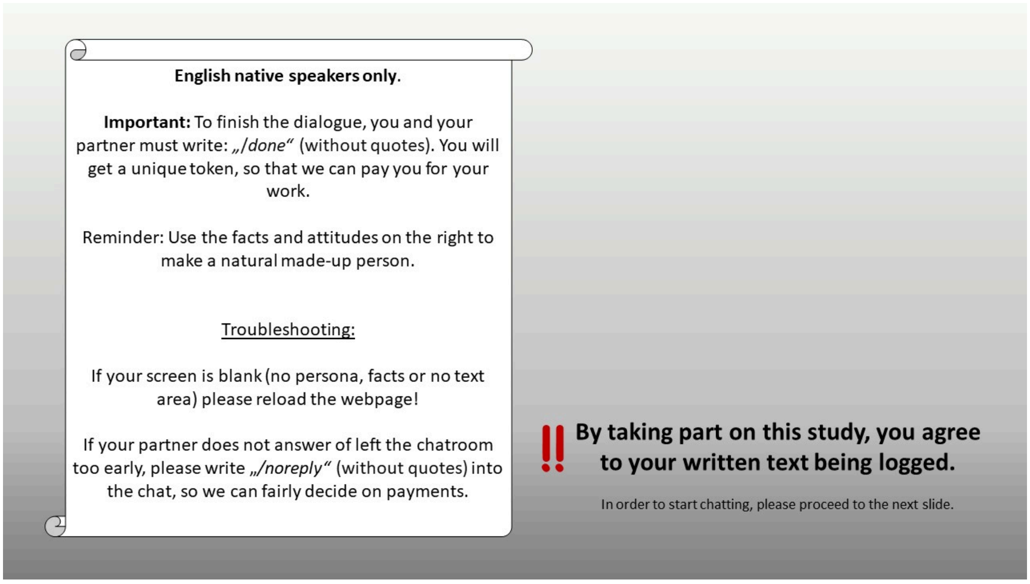

Figure A.5.: Fifth introduction slide for our data collection setup. The final slide gives some information about how the slurk setup works, beside the actual writing.

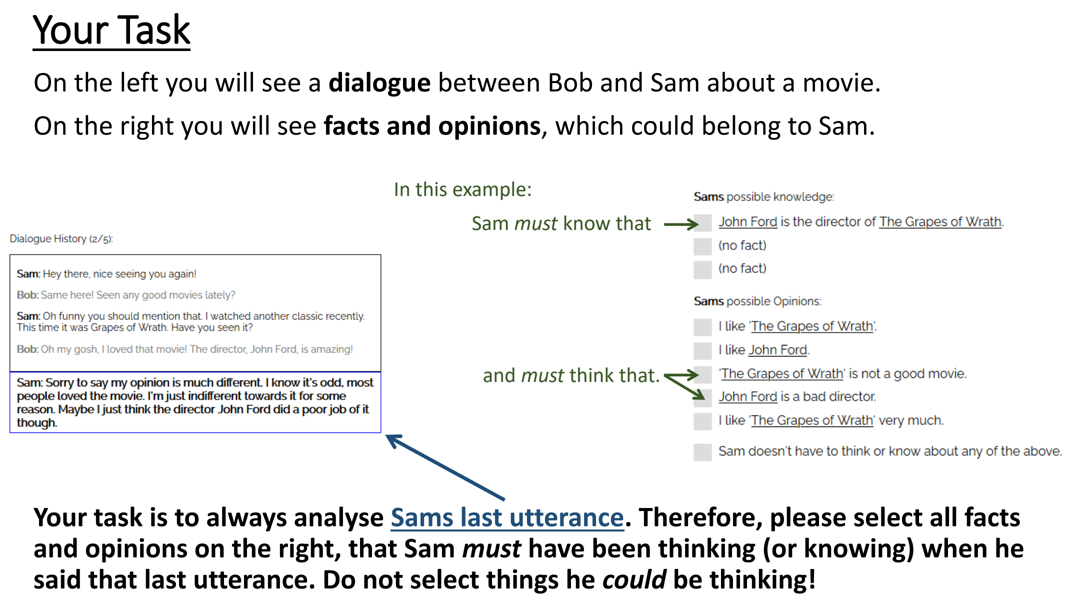

Figure A.6.: First instruction slide for crowd worker on AMT that introduces the user interface and the correct labels on the right.

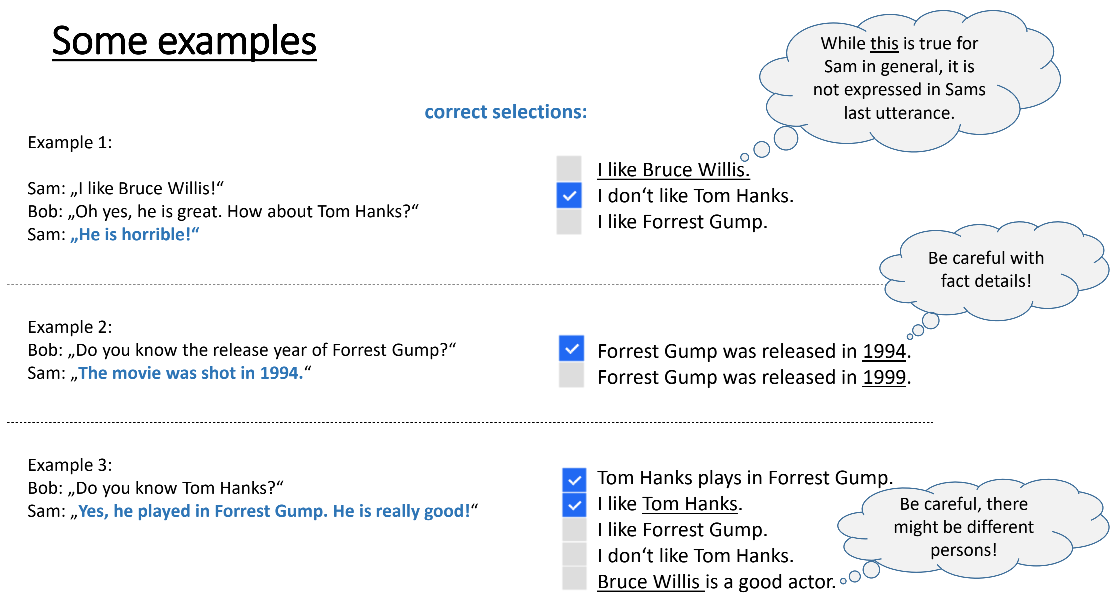

Figure A.7.: Second instruction slide for crowd worker on AMT that provides more details on how to label the data correctly.

B. Human Evaluation Questionnaire

		Dialogue	With which Person B would you prefer to talk?	Why?	Person B sounds natural	Person B sounds consistent	Person B sounds interesting
1	A	hey, how are you? I watched Sarfarosh yesterday.	2	It would be more interesting to talk to the second Person.	5	3	4
	B	i'm good thanks... yourself?. i don't know that movie actually... what's it about?					
	A	I am good! It's an action movie with sonali bendre.					
	B	oh okay cool... sounds like it might be a good one to watch with your kids.					
	A	why do you think so?					
	B	i don't really care for action movies... so maybe that's why i didn't care for this one.					
	A	what do you like?					
	B	i just don't think it's a good genre to be a part of... i don't like action movies either.					
2	A	hey, how are you?			6	4	6
	B	i'm good. i just watched a movie called "twelve monkeys". have you seen it?					
	A	yes, but I didn't like it.					
	B	really? i thought it was pretty good.					
	A	Bill Raymond is such a good actor!					
	B	i don't really like bill raymond, but he did a good job in this movie.					
	A	Do you know the release year?					
	B	i believe it was 1995. do you know what the budget was for this movie by any chance?					

Figure B.1.: Excerpt of our questionnaire from the human evaluation. Matrix from Figure B.2 is missing. Participants are tasked to decide for one dialogue, reason their decision and rate naturalness, interestingness and consistency for each dialogue on a 7-point Likert scale.

Figures B.1 and B.2 show excerpts from our questionnaire used in our human evaluations for one example dialogue pair. Participants had to fill the matrix to annotate entailments between context and dialogue, and rate the dialogues according to our given statements.

<table>
<tr><th></th><th>Dialogue</th><th>Sarfarosh is of genre Action.</th><th>Smita Jaykar has trivia: Often plays the role of the mother in Bollywood movies.</th><th>Sonali Bendre is an actor in Sarfarosh.</th><th>Sarfarosh was released in 1999.</th><th>Action is not a good genre.</th><th>Sonali Bendre is very good.</th><th>I like Sarfarosh very much.</th></tr>
<tr><td>A</td><td>hey, how are you? I watched Sarfarosh yesterday.</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2">contradiction</td></tr>
<tr><td>B</td><td>i'm good thanks... yourself?. i don't know that movie actually... what's it about?</td></tr>
<tr><td>A</td><td>I am good! It's an action movie with sonali bendre.</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2">entailment</td><td rowspan="2">entailment</td></tr>
<tr><td>B</td><td>oh okay cool... sounds like it might be a good one to watch with your kids.</td></tr>
<tr><td>A</td><td>why do you think so?</td><td rowspan="2">entailment</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2">entailment</td><td rowspan="2"></td><td rowspan="2">contradiction</td></tr>
<tr><td>B</td><td>i don't really care for action movies... so maybe that's why i didn't care for this one.</td></tr>
<tr><td>A</td><td>what do you like?</td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2">entailment</td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td>B</td><td>i just don't think it's a good genre to be a part of... i don't like action movies either.</td></tr>
</table>

Figure B.2.: Excerpt of our questionnaire from the human evaluation. Participants are tasked to evaluate utterances from person B by marking them as entailing or contradiction to each given fact or opinion individually.

C. Generated Sequences with Reinforcement Learning

The following tables show generated sequences during the reinforcement learning experiments. We only show interesting results per training to reduce the table sizes for better readability. Last step of the training is included in all tables. The differences between generated sequences within the same training did not only come from the adapting weights, but also from the random sampling that is applied during training.

step	**reply**
0	that is cool , i work as an stunt double , so i do not have a lot of time
1,000	that is interesting , i don't have any siblings .
2,000	i work at a circus as a stunt double , so i am always busy .
5,000	that is cool , i like to read . i love to read .
10,000	that is cool . do you have any hobbies ?
15,000	that is cool . do you have any hobbies ?
25,000	that is cool . do you have any hobbies ?

Table C.1.: Generated replies by a PersonaChat model trained only on fluency reward. Last utterance was: "my dad was always busy working at home depot." Relevant persona information is: "i'm a stunt double as my second job."

step	reply
0	i like winter , i want to get out of this house and away from this house .
5,000	i like to read alot . i’m a stunt double as my first job .
10,000	i like to read when i’m not working , i am trying to get a job .
25,000	i have been raised in a single parent home and i love cold weather . i like to read .

Table C.2.: Generated replies by a PersonaChat model trained only on consistency reward. Last utterance was: “wow ! i do love a good horror movie . loving this cooler weather” Relevant persona information is: “i’m a stunt double as my second job.”, “i read twenty books a year .” and ‘´i was raised in a single parent household .”

step	reply
0	that is cool , i am in a stunt double , it is my favorite job .
5,000	i’m studying for my third job , i’m a stunt double
10,000	that’s pretty cool . i like horror .
25,000	i love horror movies

Table C.3.: Generated replies by a PersonaChat model trained only on repetition reward. Last utterance was: “i just got done watching a horror movie.” Relevant persona information is: “i’m a stunt double as my second job.”

step	reply
0	i love to work on vintage cars .
5,000	i have to work on vintage cars , i love to work on vintage cars
6,000	that is horrible . i love to work on vintage cars .
10,000	i work at a vintage car repair shop .
25,000	that is terrible . i used to be a homeless person , now i work at a vintage car repair

Table C.4.: Generated replies by a PersonaChat model trained only on repetition reward. Last utterance was: “i work in a homeless shelter in my town .” Relevant persona information is: “i like to work on vintage cars .” and “i own two vintage mustangs.”

step	reply
0	hi, no, i haven't. what's it about?
500	yes i have seen it and i think it is my favorite movie ever!
2,500	no, i haven't. what is it about?
6,000	yeah i've seen it! what was the release date?
12,500	yeah! what was the movie about? what was the age limit?
13,000	no! what was it about? what was the release date?

Table C.5.: Generated replies by a KOMODIS model trained only on repetition reward. Last utterance was: "hey! have you ever seen indiana jones? the last crusade?" Relevant context is: "Indiana Jones and the Last Crusade is my favorite movie."

step	reply
0	i agree with the age restriction
500	harrison ford is my favourite actor
3,500	damn! harrison ford plays in indiana jones. i think his performance was great
10,000	damn, harrison ford isn't my favorite actor
10,500	yeah, harrison ford is my favourite actor
13,000	damnnhhh, harrison ford is my fav actor

Table C.6.: Generated replies by a KOMODIS model trained only on repetition reward. Last utterance was: "it is one of mine as well! except, i think it would be better with out harrison ford, he is not a very good actor." Relevant context is: "Harrison Ford is my favorite actor."

step	reply
0	i agree with the age restriction, but i think the age restriction is a bit high.
5,000	oh, i didn't realize. so why would you want to see it?
10,000	oh, i thought it had decent lighting!
13,000	that is fair. anyway, i must go for now. take care!

Table C.7.: Generated replies by a KOMODIS model trained only on repetition reward. Last utterance was: "i agree.. that seems appropriate. i am surprised it did not win best picture you know."