FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Vojtěch Hudeček

Low resource methods for dialogue systems applications

Institute of Formal and Applied Linguistics

Supervisor: Mgr. et Mgr. Ondřej Dušek, Ph.D.

Study Program:	Computer Science
Specialization:	Computational Linguistics

Prague 2023

Title: Low resource methods for dialogue systems applications

Author: Vojtěch Hudeček

Department: Institute of Formal and Applied Linguistics

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

Abstract:

This thesis focuses on developing and improving task-oriented dialogue systems design in the rapidly growing landscape of artificial intelligence and natural language processing. We propose techniques that can substantially decrease development and deployment costs, motivated by the desire to make these systems more adaptable and scalable. We introduce multiple novel approaches to achieving these goals.

Firstly, we present a weakly supervised automatic data annotation pipeline that can transform raw dialogue transcript into a refined set of semantically coherent concepts, bypassing the need for exhaustive manual annotations in natural language understanding for a given domain and significantly streamlining the development process.

We also explore the largely uninvestigated field of latent variable models in task-oriented dialogue system modeling. These models offer excellent capabilities with the potential to uncover the structure of behavioral patterns seen in the dialogue through inspection of the latent space and comparison with actions taken by the model. Furthermore, we explore the potential of these models to form hierarchical representations using our proposed architecture.

Following recent progress in the field, we harness the power of pre-trained large language models using in-context learning. We explore how easily language models can transfer the learned knowledge to previously unseen domains. We also propose a method based on retrieval-augmented LLM prompting that performs well with merely a few training examples. It shows great promise in our human evaluation trial, implying a substantial leap in efficiently using computational resources to train conversational AI. This brings us closer to more flexible and general-purpose systems.

Keywords: dialogue systems, semi-supervised learning, language models, variational training, hierarchical models, clustering

Název práce:	Metody pracující s omezeným množstvím zdrojů pro využití v dialogových systémech
Autor:	Vojtěch Hudeček
Katedra:	Ústav formální a aplikované lingvistiky
Vedoucí práce:	Mgr. et Mgr. Ondřej Dušek, Ph.D.

Abstrakt:

Tato práce se zaměřuje na vývoj a zdokonalování návrhu tzv. task-oriented dialogových systémů v rychle se rozvíjejícím prostředí výzkumu umělé inteligence a zpracování přirozeného jazyka. Navrhujeme techniky, které mohou podstatně snížit náklady na vývoj a nasazení těchto systémů, což je motivováno snahou o jejich větší přizpůsobivost a škálovatelnost. V práci představujeme několik nových přístupů k dosažení těchto cílů.

Nejdříve představujeme automatickou metodu anotace dat, která dokáže extrahovat sadu sémanticky koherentních konceptů (dialogových slotů) z prostého přepisu zaznamenaných konverzací. Tímto přístupem snižujeme množství manuální anotace potřebné pro porozumění přirozenému jazyku v dané doméně a výrazně tak zefektivňujeme proces vývoje.

Zkoumáme také modely využívající latentní proměnné v modelování task-oriented dialogových systémů. Tato oblast je do značné míry neprobádána. Modely využívající latentní proměnné nabízejí možnost využití neanotovaných dat s potenciálem odhalit strukturu vzorců chování pozorovaných v dialogu. Toho lze dosáhnout prostřednictvím analýzy latentního prostoru a porovnání s akcemi provedenými modelem. Dále zkoumáme potenciál těchto modelů pro vytváření hierarchických reprezentací pomocí námi navržené architektury.

V návaznosti na nedávný pokrok v této oblasti také využíváme schopnosti předtrénovaných velkých jazykových modelů (LLM) pomocí metody tzv. in-context learning, tedy učení se z kontextu. Zkoumáme, jak snadné je pro jazykové modely aplikovat znalosti získáné tréninkem v jedné doméně na dříve neviděných datech. Námi navržená metoda založená na učení z kontextu obohaceném o příklady dosahuje pozitivních výsledků s použitím pouze několika tréninkových příkladů. Ukázala se také jako velmi slibná při hodnocení v interakcích s lidmi. Tento způsob použití modelů představuje podstatný skok v efektivním využívání výpočetních zdrojů k trénování konverzační umělé inteligence. To nás přibližuje k flexibilnějším a univerzálnějším systémům.

Klíčová slova: dialogové systémy, semi-supervised learning, jazykové modely, hierarchické modely, clustering, variační autoenkodéry

Acknowledgements

First, I would like to thank my supervisor, Ondřej Dušek, who guided me through my studies, inspired me with ideas, tirelessly reviewed my work, and was always willing to help in any way.

A huge thank you goes to my parents, Jan and Pavla, and brother Štěpán, who raised me to be curious and persistent and supported me in many ways.
Thank you to all my friends, too many to name here, who always cheered me up and motivated me. Whether you knew it or not, many conversations with you kept me going.

I want to express my gratitude to my colleagues from the Institute of Formal and Applied Linguistics for their enthusiasm, consultations, and feedback, which was greatly appreciated. Thanks, Ondřej Plátek, Jakub Náplava, Zdeněk Kasner and many others.
I also appreciate those who trusted me and offered internship opportunities or collaborations – Zhou Yu, Alex Papangelis, Mahdi Namazifar, and Zdeněk Žabokrtský.

I couldn't have done this without all of you.

Díky, Vojcek

This work was supported by the European Research Council (Grant agreement No. 101039303 NG-NLG), and by the Charles University project SVV 260575. It used resources provided by the LINDAT/CLARIAH-CZ Research Infrastructure (Czech Ministry of Education, Youth and Sports project No. LM2018101). Parts of the work were also supported by Charles University grants PRIMUS 19/SCI/10, GAUK 302120, project TL05000236 *AI asistent pro žáky a učitele* co-financed by the Technological Agency of the Czech Republic within the ÉTA 5 Programme and the ANRT CIFRE #2017/1543, HumanE-AI-Net project / EC Horizon 2020, Grant Agreement H2020-FETFLAG-2018-2020 no. 952026

Contents

1 Introduction

Human language is a convenient and natural means of communication for human beings. It is, therefore, desirable to implement an interface that mimics natural language and allows humans to interact with computers like they would with other human individuals.

To achieve this goal, we need to be able to transfer information between human users and the computer. Humans most often use speech or writing to encode and transfer information, so various techniques have been invented that deal with this kind of encoding, such as Automatic Speech Recognition (ASR), Optical Character Recognition (OCR), and Text-to-speech Synthesis (TTS). However, to efficiently transfer information, we need the ability to engage in a conversational exchange. A conversation (dialogue) offers additional means of communication such as clarification, information updates, or more effective encoding through context reference, etc.

To perform meaningful dialogue, we need more than just mimicking the interface. The computer should understand the process of gradual information exchange and be able to capture the meaning of utterances in the context. Moreover, the system must provide relevant responses to engage in the conversation successfully(Jurafsky, 2000; McTear, 2022).

In this work, we focus on this part of the problem, i.e. we do not care about encoding or decoding natural language in a signal such as speech. Rather, we assume textual interfaces for both input and output. Put simply, the task of a Dialogue System (DS)(Jurafsky, 2000) is to generate the correct natural language response r given the natural language user utterance u and context c. Importantly, it is not always clear what we mean by a “correct utterance” in a particular context. This can depend on several conditions, requirements, and constraints. In this work, we consider task-oriented dialogues, which are well-defined in this aspect. This concept is introduced in more detail in Section 2. We understand the dialogue as a *turn-taking* conversation, i.e. participants (user and system) communicate in alternating *turns*. In this work, we exclusively focus on two-party dialogues.

Considering the conversation history, the ultimate goal is to construct a dialogue agent that provides meaningful responses to all kinds of questions. Such an agent would effectively pass the Turing test, the holy grail for Artificial Intelligence (Pinar Saygin et al., 2000). The development of Large Language Models (LLMs) and their instruction tuning brings us closer to achieving this goal (Rothman, 2021). Nevertheless, we do not need to achieve such complexity in many real-life cases. For example, we can consider situated artificial agents which solely focus on achieving a certain well-specified goal, such as ordering food or reserving a flight ticket.

Dialogue systems promise a convenient means of communication between humans and computers. They allow voice interaction, making it especially well suited for applications that should not disrupt attention, such as car control. Systems capable of human-like conversation and accomplishing given tasks have huge potential to automate technical support processes and call centers or serve as personal assistants.

Despite some successful dialogue system deployments, current dialogue systems still suffer from several drawbacks. Usually, the DSs are tailored to specific applications, and applying them in other domains is hard. Typically, the system is customized to handle a set of predefined domains with a high success rate. A lot of effort goes into designing an ontology and handling domain-specific scenarios. Even in the era of large language models, this is still the case for the predominant part of commercial applications. This results in bad scalability and inflexible usage. Ideally, a system would learn common behavioral patterns required to finish the defined goal through conversational exchange successfully. Given some description data, it could apply the learned knowledge to previously unseen domains and applications. Although the LLMs make a huge step forward in this ability, they still might require finetuning and are not yet suitable for direct applications in the task-oriented world (Iizuka et al., 2023; Hudeček and Dušek, 2023) which we also discuss in Chapter 7.

Another problem is that there seems to be a trade-off between interpretability and performance or scalability of the systems in the case of neural network-based models. In most cases, the more complex and capable the model is, the harder it is to interpret its behavior and explain its decisions.

This thesis proposes solutions to some of these problems, especially in the task-oriented setting. We now outline the main goals we want to achieve and the ways we address them in our experiments:

1. **Limit the amount of supervision needed**. Extending the system is hard since it requires significant expert effort to design the schemas and annotate the data. In Chapter 4, we propose an automatic data analysis tool to gather information from dialogue corpora and suggest annotation schema without direct supervision.

2. **Enable the dialogue systems to leverage large unannotated data sets and train more robust models**. There has not been much work on training task-oriented systems in an unsupervised way. We delve into this problem and suggest using latent variable models in Chapter 5 and explore the usage of pre-trained language models, which leverage large unannotated corpora, in Chapters 6, 7.

3. **Be able to train the systems with less data overall** It is difficult for the current system architectures to transfer the learned knowledge to new domains. To explore this phenomenon more, we conduct experiments to see if language models can transfer the knowledge in Chapter 6. We also explore how to train LLMs with a limited number of examples in Chapter 7.

Scalability and domain adaptation go hand in hand. We focus on reducing the annotation needed to train a system and on knowledge abstraction to make transfer learning possible. To leverage larger data sets, we explore unsupervised techniques that do not require annotation, making the data collection process substantially easier.

In addition to the experimental chapters mentioned above, this thesis also includes Chapters 2,3, which introduce the theoretical concepts and related from which we take inspiration or use for comparison. Finally, Chapter 8 summarizes our findings and proposes future research directions.

2 Background

Before we introduce our work and delve into discussions, we will establish the common theoretical background of the concepts we use. Some of the concepts are well known, some less, but we consider it important to have this reference point and provide at least a concise description. First, we briefly describe this work's basic concepts and foundational models in Section 2.1. We link to literature when needed. Next, we introduce methods from related research areas upon which we build in our experimental chapters. Specifically, we first discuss pre-trained language models (Section 2.2). Next, we introduce variational autoencoders (Kingma and Welling, 2014) and some architectures that build upon them, such as variational recurrent neural networks (Chung et al., 2015) in Section 2.3. We also describe the memory network architecture principal ideas (Weston et al., 2015) in Section 2.4. Next, Section 2.5 describes some typical approaches to dialogue system implementations. Ultimately, we introduce the datasets we use in this work for training and testing our models and evaluation metrics we are working with in Sections 2.6.1 and 2.7 respectively.

2.1 Key concepts

In this section, we first introduce some key concepts necessary for understanding this work and establish common ground.

2.1.1 Neural Networks

The nature of many Natural Language Processing (NLP) tasks is statistical in that we can identify patterns seen in data and aim to model the underlying processes. Therefore, statistical methods in NLP have been extensively used for decades (Manning and Schütze, 1999), with varying success. Although many modeling approaches with various degrees of complexity have been proposed, currently, Neural Network models dominate the field. Neural architectures are well-known statistical models (Goodfellow et al., 2016) that can learn complex data distributions from the presented training data. Essentially, these models parameterize non-linear functions with millions of parameters learned by optimizing loss objective over the training set using algorithms such as backpropagation (Kelley, 1960).

Neural Networks for text processing Neural Networks (NN) are widely used across various fields, and NLP is among the most prominent. However, it is not straightforward how to represent words in NN. A simple one-hot vector or bag of words representation is insufficient as it throws away a lot of useful information about the relations between words, their position, etc. Traditional statistical n-gram models (Jurafsky, 2000) address some of the issues but again discard some positional information and are inherently limited by context size (n-gram length). A breakthrough in this field, which allowed for successful NN applications, dates back to the works of Mikolov et al. (2010) and Mikolov et al. (2013), which build on the principle of word distributional hypothesis and introduced the concept of *word embeddings*. Word embeddings are multi-dimensional real vectors that capture important information about each word in some particular input corpora. When input to an NN, word embeddings accurately represent the word and arguably capture the meaning. This principle was later improved by introducing large-scale pre-trained neural language models such as ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019b), which can create contextualized high-dimensional representations. We introduce these principles in more detail in Section 2.2.

2.1.2 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks are a type of neural network well-suited for processing sequential data (Goodfellow et al., 2016). Sequential data is ordered in time, such as text or audio. RNNs can learn long-range dependencies between different parts of the input sequence, which makes them well-suited for tasks such as machine translation, text summarization, and question answering. The motivation for RNNs comes from human language being sequential by nature; therefore, sequential dependencies must be handled.

The characteristic feature of an RNN is a recurrent connection incorporated into the architecture. This means that the network's output at a given time step is fed back into the network at the next step. This allows the network to encode some information and pass it for future processing, effectively allowing for a concept of memory.

Basic RNN consists of 3 weight matrices: W_i to process the input, W_h to process the hidden state, and W_o to construct output. The computation of output sequence $\mathbf{y} = \{y_0, ..., y_n\}$ from an input sequence $\mathbf{x} = \{x_0, ..., x_n\}$ proceeds in the following steps:

1. An initial hidden state h_0 is constructed.
2. For time step t, the first hidden state h_t is obtained using the formula $h_t = f(W_h h_{t-1} + W_i x_t)$. Then, the output is computed as $y_t = W_o h_t$.

Note that the output sequence has the same length as the input sequence, which is useful for sequence tagging problems. However, RNNs can also be used for sequence classification, in which case only the last hidden state is considered, or the encoder-decoder setup can be used in which one network is used to encode the sequence. The second one generates the output, which can be of different lengths.

RNN cell improvements Vanilla RNNs are difficult to train and often fail to memorize long-term dependencies correctly. Therefore, several extensions were proposed such as LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014) These more complex types of RNNs can learn long-range dependencies more effectively. LSTM and GRU cells have several gates that control how information flows through the cell. These gates allow the RNN cell to learn how to forget irrelevant information and remember important parts of the input. NLP models often employ *bidirectional* RNNs to improve modeling capabilities further, allowing for processing both left and right context.

Training RNNs RNNs are trained using a technique called Backpropagation Through Time (Werbos, 1990). Backpropagation Through Time is a method for training neural networks that deal with sequential data. The basic idea is to break the sequence into several time steps and then train the network on each time step individually.

Encoder-Decoder RNNs with attention The encoder-decoder RNN is useful for text-generation tasks like summarization or machine translation. However, it isn't easy to train RNN on longer sequences. Therefore, improvements were proposed (Bahdanau et al., 2014; Luong et al., 2015) that allow the decoder network to attend to the input sequence and bypass the need to memorize all the information in the hidden state.

2.1.3 Transformer

The Transformer architecture is another neural network architecture capable of processing sequential data. It was first proposed in Vaswani et al. (2017). Unlike RNNs, the Transformer architecture does not work with a hidden state that is being passed forward in time. Instead, the architecture is based solely on the attention mechanism, which allows the model to learn long-range dependencies between different parts of the input sequence. The high-level equation describing the self-attention mechanism is formulated as follows:

$$Att(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}V\right)$$

where Q, K and V are trainable square weight matrices corresponding to *queries*, *keys* and *values* and with dimension d_q, d_k, d_v respectively. The attention mechanism allows the model to focus on different parts of the input sequence when producing the output sequence. For example, if the model is translating a sentence from English to German, the attention mechanism can allow the model to focus on the English words most relevant to the German words that need to be produced.

The original Transformer architecture comprises an encoder and a decoder, depicted in Figure 2.1. The encoder takes the input sequence and produces a sequence of hidden representations. The decoder then takes these hidden representations and produces the output sequence. The attention mechanism is used at both the encoder and decoder to allow the model to use different parts of the input sequence when producing the output sequence. The decoding process might be autoregressive or non-autoregressive, depending on the use case.

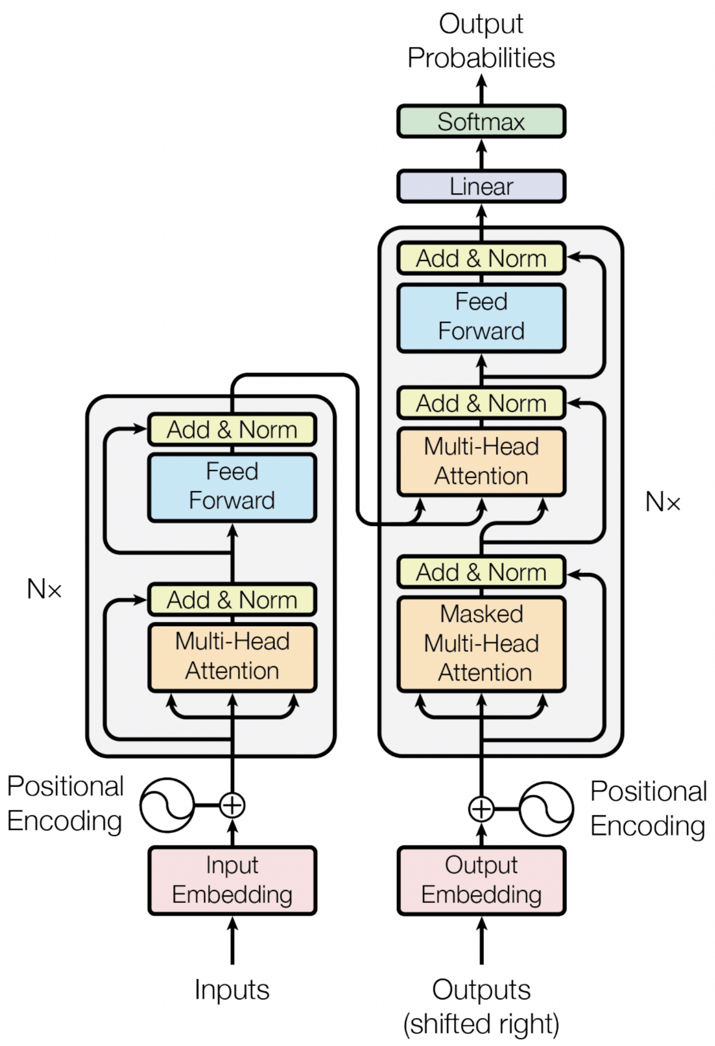


Figure 2.1: The Transformer architecture as proposed by Vaswani et al. (2017). We can see the encoder (left) and decoder (right) stacks and their interconnection.

Both the encoder and decoder are a stack of self-attention layers. Each self-attention layer takes the hidden representations from the previous layer and produces a new set of hidden representations.

The decoder also includes an attention component that allows the model to attend to the output sequence generated so far when producing the next token. This allows the model to learn how to generate the output sequence consistently with the input sequence and the prefix generated so far.

The Transformer architecture has been shown to achieve state-of-the-art results on various natural language processing tasks. Moreover, the Transformer architecture was used as a base for many models pre-trained on large natural language corpora, allowing efficient transfer of the learned information to downstream tasks and thus revolutionizing the NLP field.

2.2 pre-trained Language Models (PLMs)

The appearance of PLMs marked a new era in NLP. Although Language Modeling with neural networks has been in the community focus for some time with transforming works such as Mikolov et al. (2010, 2013), it was not until RNN-based models such as ELMo (Peters et al., 2018) and ULMFiT (Howard and Ruder, 2018) were proposed that we were able to create PLMs that could be fine-tuned for a variety of tasks with comparably low data requirements.

Shortly after ELMo, models based on the Transformer architecture started to appear, such as BERT for efficient language encoding (Devlin et al., 2019b) or GPT for generation (Radford et al., 2018). BERT introduced a novel approach to Transformer usage by employing only the encoder part of the original Transformer architecture. Importantly, BERT pre-training introduced the task of Masked Language Modeling, randomly masking some of the input tokens and reconstructing them correctly at the output. Additionally, BERT is trained to estimate the probability that two input sentences follow each other. These pre-training techniques make BERT great at encoding natural language inputs into useful representations. GPT, on the other hand, consists only of Transformer decoder blocks. It is trained for next token prediction and can perform autoregressive Language Modeling. Therefore, GPT is a perfect candidate for a base model to be fine-tuned on language generation tasks, including data-to-text, summarization, etc.

Both BERT and GPT showed great potential for fine-tuning downstream tasks and established themselves as strong baselines for many NLP tasks and benchmarks. They are often referred to as *foundational* models because of their language understanding and generation abilities and the potential to apply these abilities to various tasks. From a certain point of view, the usability of PLMs lies in the ability to create useful and contextual representations of input words and phrases, often called *(word) embeddings.* This ability has been observed in the word2vec model (Mikolov et al., 2013) and improved by ELMo (Peters et al., 2018), BERT (Devlin et al., 2019b), SBERT (Reimers and Gurevych, 2019b) and others.

Many model architectures based on Transformer blocks followed both for encoders (Liu et al., 2019; Reimers and Gurevych, 2019b); decoders (Radford et al., 2019; Brown et al., 2020) or even encoder- decoder (Raffel et al., 2020b; Lewis et al., 2020). The NLP community largely adopted these models.

2.2.1 Large Language Models (LLMs)

The foundational models changed the NLP paradigm but still need substantial in-domain data to perform some tasks well. Moreover, fine-tuning those larger models requires more computational resources. Therefore, more lightweight methods were proposed to adapt the models to downstream tasks, such as Transformer Adapters (Pfeiffer et al., 2020). However, as researchers started to scale up the models with GPT-2 and GPT-3 leading the efforts (Radford et al., 2019; Brown et al., 2020), new abilities emerged (Wei et al., 2022). With model sizes exceeding billions of parameters, the large pre-trained Transformer decoders can perform many tasks not explicitly trained for (Brown et al., 2020). Such large models can perform tasks like summarization, translation, question answering, or even reasoning and arithmetics to some extent without any task-specific training. However, it is unclear how many examples of the abovementioned tasks were seen during these models' training phase. Those tasks can be presented to the LLMs in textual task descriptions in the inference time. This approach of *in-context learning* is frequently used with great success (Min et al., 2022; Dong et al., 2022). The textual input for LLMs is frequently called a *prompt*.

On LM scaling Scaling laws in language models describe the relationship between the size of a language model and its performance. It has been shown that with the next token prediction objective (which virtually all LMs are trained for), some abilities emerge when exceeding a certain size threshold (Kaplan et al., 2020). However, bigger models must also be trained longer and with more data (Hoffmann et al., 2022). In general, larger language models tend to perform better than smaller models but also require more computational resources to train. This line of research can help us understand how the size of a language model affects its performance and predict the performance of a language model before it is trained.

Instruction Tuning Although the LLMs have great abilities and potential to accomplish various tasks, providing them with correct instructions is not always straightforward. Therefore, significant effort was made to *align* the LLMs better with the human requirements. Consequently, even rather inexperienced users can instruct the model to accomplish custom tasks according to their needs. For this purpose, reinforcement learning techniques were explored (Ziegler et al., 2019; Ouyang et al., 2022). Although these techniques proved to be quite effective, the process is still very demanding in collecting user feedback. Consequently, several datasets were proposed (Wang et al., 2022; Black et al., 2022) that contain tasks like summarization, reasoning, etc. formulated using instructions in natural language and desired outputs. These datasets allow tuning of the models using reinforcement learning or supervised fine-tuning methods.

Prompt engineering vs. LLM fine-tuning The in-context learning approach has great advantages because it makes it possible to obtain great performance from LLMs by simply formulating the task and desired output structure in the LLM input (i.e. prompt). There are multiple strategies to formulate the prompt of the in-context learning technique. We can include examples for better model guidance or just formulate the instructions. These approaches are referred to as *few-shot* or *zero-shot* settings, respectively. Although this approach might be very efficient for some tasks, especially those that are well described in the corpora available to the LLM during training (Wei et al., 2022), some more specific tasks might yield better results after fine-tuning of the model (Tu et al., 2022). However, the LLM fine-tuning process is quite demanding regarding computational resources. Therefore, alternative approaches were proposed, such as LoRA (Hu et al., 2021) or Transformer Adapters (Pfeiffer et al., 2020). These approaches make LLM fine-tuning much more accessible. In general, fine-tuning and in-context learning can offer great performance and be beneficial in certain situations (Mosbach et al., 2023).

2.3 Variational autoencoders

In neural network training, the network learns to create internal data representations to accomplish a given task. In the case of autoencoders, the task is to encode an input $\mathbf{x}$ in a way that allows for its reconstruction into the original form. The autoencoder model consists of an encoder function φ^{enc}, which encodes an input $\mathbf{x}$ into a latent representation $\mathbf{z}$, and a decoder φ^{dec}, which models the conditional re-generation probability $p(\mathbf{x}|\mathbf{z})$. In the case of sequence autoencoders, both the encoder and decoder can be realized with an RNN. However, vanilla autoencoders often fail to extract global semantic features of natural language sequences (Bowman et al., 2015); therefore, adjustments must be made to obtain better representations. The technique proposed by Kingma and Welling

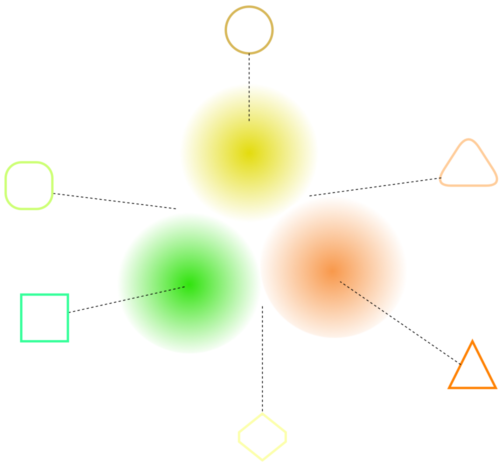


Figure 2.2: Variational autoencoder latent space. Colors and different classes by shapes distinguish the encoder distributions. It illustrates that we can smoothly interpolate between two points in the latent space.

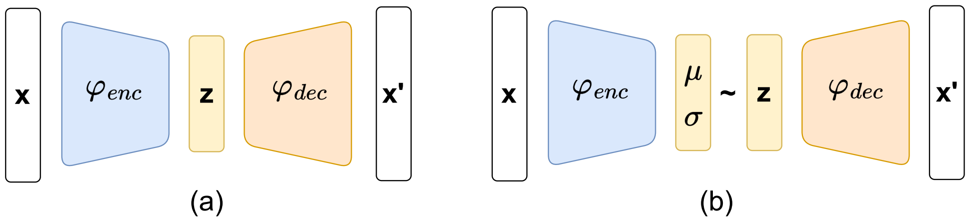


Figure 2.3: Illustration of differences between the architectures of a vanilla autoencoder (a) and its variational version (b). The VAE encodes the input by predicting parameters of probabilistic distribution from which the data are drawn rather than encoding the data directly into a hidden representation, as done with the vanilla autoencoder.

(2014) uses the Variational Autoencoder (VAE) framework to tackle this issue. The architecture is modified so that φ^{enc} represents a recognition model $q(\mathbf{z}|\mathbf{x})$ which parameterizes an approximate posterior distribution over $\mathbf{z}$. Figure 2.3 illustrates the differences. VAEs impose prior distribution on the latent variable $\mathbf{z}$, which acts as regularization during training and makes drawing samples from q possible. Consequently, the VAE latent space is smooth because it is possible to interpolate between two points and obtain reasonable representations. The latent space structure is depicted schematically in Figure 2.2. The modeled distributions are typically Gaussian; the prior is the standard normal distribution $N(0, 1)$. We can realize the encoder and decoder modules in VAE using neural networks. However, there is a drawback regarding the implementation of sampling. The sampling operation is not differentiable and, therefore, cannot be trained using standard approaches. A solution to this problem is to use the *reparameterization trick* (Kingma and Welling, 2014). The reparameterization

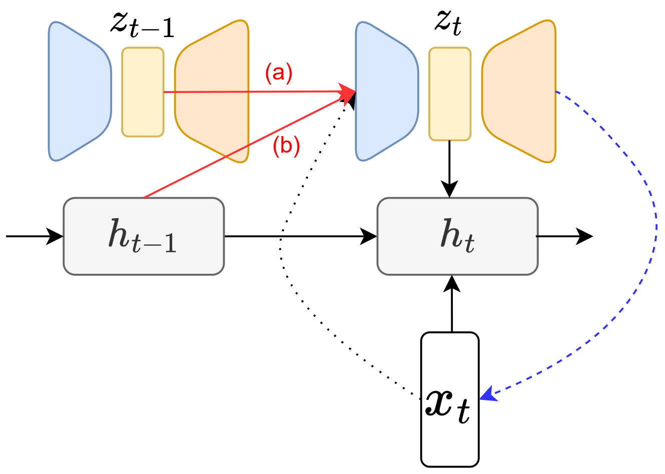


Figure 2.4: A schematic architecture of the VRNN model. The input in time step t is input to the variational autoencoder (black dotted line). The VAE prior is conditioned either on the previous hidden state (a) or previous latent variable z_{t-1} (b) (red lines). To regenerate the output, the decoder from latent space is used (blue dashed line). Finally, the hidden state update is based on latent representation z_t, previous hidden state, and current input (solid black lines).

trick exploits that a random variable under a certain conditional distribution can be expressed as a deterministic transformation of some other variable with an independent marginal distribution. Distributions that allow us to do such a transformation include *Gaussian, Logistic*, or *Gumbel* (Jang et al., 2016).

VAE latent space discretization

Although VAE training yields robust representations that are also more interpretable thanks to the regularized latent space, in some cases, we require the latent representations to be discrete. The motivation is mainly to improve interpretability and uncover underlying processes in sequential tasks. Incorporating discrete variables into neural network models is problematic because the widely used backpropagation algorithm requires smooth differentiable functions to propagate the gradients correctly. van den Oord et al. (2017) propose a vector quantization technique to discretize the latent variables in VAEs. Another approach is to use the Gumbel-softmax distribution (Jang et al., 2016) that, together with the reparameterization trick (Section 2.3), enables working with categorical variables while not breaking the gradient flow in the network.

2.3.1 Variational Recurrent Neural Networks

The VRNN model (Chung et al., 2015) exploits the idea of variational training to model sequences with latent states. Intuitively, the VRNN model can be seen as a recurrent network with a VAE in every timestep. The VRNN architecture is depicted in Figure 2.4. It assumes that the sequence of observations was generated from a sequence of unknown latent states and uses a VAE to model these latent variables $\mathbf{z}$. Formally, we want to estimate the joint probability distribution of a sequence of observations and corresponding latent variables $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$. The conditional distribution $p(\mathbf{x}|\mathbf{z})$ is parameterized with a neural network. However, we still need to estimate the posterior $p(\mathbf{z}|\mathbf{x})$ to connect the latent variables with the observations. The VAE uses a variational approximation $q(\mathbf{z}|\mathbf{x})$ that allows maximizing the evidence lower bound (ELBo) of the log-likelihood of the data:

$$\begin{aligned} \log\ p(\mathbf{x}) \geq & -\mathrm{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \\ & + \mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log\ p(\mathbf{x}|\mathbf{z})] \end{aligned} \tag{2.1}$$

where KL is the Kullback-Leibler divergence.

We consider a prior network φ_{prior} and a posterior network φ_{post}, which compute the parameters of $p(\mathbf{z})$ and $q(\mathbf{z}|\mathbf{x})$ respectively. In a VRNN, φ_{prior} and φ_{post} additionally depend on the RNN hidden state $\mathbf{h}^t$ to allow for a context-aware prior distribution. In each time step, we obtain the distribution parameters as follows:

$$\begin{aligned} \theta_q &= \varphi_{post}(\mathbf{h}^t, \varphi_{enc}(\mathbf{x}^t)) \\ \theta_p &= \varphi_{prior}(\mathbf{h}^t) \end{aligned} \tag{2.2}$$

where φ_{enc} is the encoder and θ_q, θ_p are parameters of the respective distributions. With distribution parameters available, we can sample the latent variable and predict the output:

$$\begin{aligned} \mathbf{z}^t &\sim p(\mathbf{z}; \theta_p) \\ \mathbf{x}^t &= \varphi_{dec}(\mathbf{z}^t) \end{aligned} \tag{2.3}$$

where φ_{dec} represents the decoder network. The update of the hidden state $\mathbf{h}^t$ is as follows:

$$\mathbf{h}^{t+1} = \mathrm{RNN}([\varphi_{enc}(\mathbf{x}^t), \varphi_z(\mathbf{z}^t)], \mathbf{h}^t) \tag{2.4}$$

where $[.,.]$ is concatenation, $\varphi_z(.)$ is a feature extractor, and RNN() is a step transition function of a recurrent neural network, in our case an LSTM (Hochreiter and Schmidhuber, 1997).

2.3.2 Latent Action Spaces via Variational Auto-encoding – LAVA

The LAVA framework (Lubis et al., 2020) focuses on learning latent variables in a way that they store dialogue-related actions. To achieve this, they employ VAE-based model architecture. The architecture comprises an RNN utterance encoder, VAE module, and RNN decoder. They train the model in multiple stages and different variants. Specifically, they first pre-train the encoder module using an autoencoding objective. Then, they use the pre-trained decoder and train an additional encoding module for the next response prediction. In the end, they use reinforcement learning to tune the model parameters further.

Formally, the authors first train an encoder-decoder network with an autoencoding objective version of ELBo parameterized by ϕ.

$$\mathbb{L}_{ae}(\phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log\ p_\phi(\mathbf{x}|\mathbf{z})] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{2.5}$$

where $\mathbf{x}$ represents dialogue utterance. Then, they train the response generation network parameterized by θ using the lite ELBo objective (Lubis et al., 2020).

$$\mathbb{L}_{lite}(\theta) = \mathbb{E}_{q_\theta(\mathbf{z}|\mathbf{c})}[\log\ p_\theta(\mathbf{x}|\mathbf{z})] - \beta\mathrm{KL}(q_\theta(\mathbf{z}|\mathbf{c})||p(\mathbf{z})) \tag{2.6}$$

where $\mathbf{c}$ is a sequence of utterances representing the dialogue context. For the second step, the decoder network p_θ is initialized by pre-trained p_ϕ.

2.3.3 Difficulties of the VAE training

For text VAEs (Bowman et al., 2016), the KL divergence term measured between the posterior and prior distributions often tends to zero. Consequently, the modeled latent space degrades, and the latent variables do not contain useful information. This phenomenon is called posterior collapse, where the model ignores the latent variables and solely focuses on the decoder for maximum likelihood estimation. To address this issue, several modifications were proposed. A common approach is to use *warm-up* or *annealing schedules* (Fu et al., 2019). In this method, the weight of KL divergence in the loss function starts at zero and gradually increases to its maximum value over several epochs. This allows the decoder to learn useful information before the encoder starts learning the prior. Another method is called *free bits* (Li et al., 2019). This method decouples the optimization of the likelihood and the KL term. Each dimension in the latent space can use a fixed capacity (bits) to encode the data before the KL term is minimized.

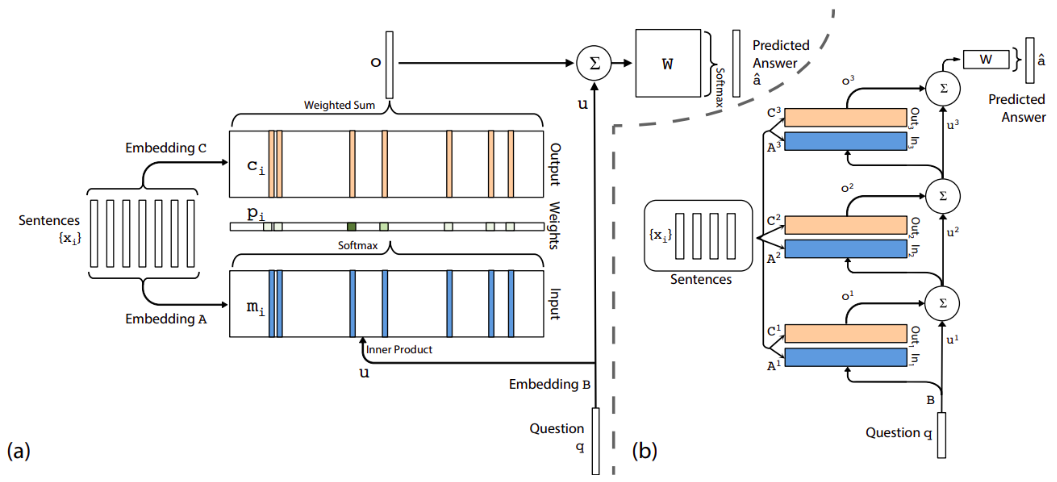


Figure 2.5: The end-to-end Memory Network model introduced in Sukhbaatar et al. (2015). It shows the computation process of (a) a single-layer and (b) a 3-layer model version.

2.4 Memory Networks

The Memory Network model (MemNN) addresses the issue associated with RNN networks, which typically suffer from catastrophic forgetting and exponential decay of stored information (Neil et al., 2017). The Memory Network model was originally introduced in Weston et al. (2015).

2.4.1 The Memory Network architecture

The key idea behind memory networks is to incorporate an external memory component into the network architecture, allowing the model to store and access information over long sequences of inputs. The memory component acts as a separate storage module, similar to a computer's memory, which the network can read from and write to during its computation. The MN model architecture's main component is a memory array $\mathbf{m} = \{\mathbf{m}_1, ..., \mathbf{m}_n\}$. The basic Memory Network takes an input $\mathbf{x}$ and processes it in four processing steps:

1. The input x is processed with *input feature map* I to obtain internal representations of the input $I(x)$.

2. The memory array is updated using the *generalization component* G. In this step, each memory entry gets updated following the equation $\mathbf{m}_i = G(\mathbf{m}_i, I(\mathbf{x}), \mathbf{m})$.

3. The output features are computed with the *output feature map* using the memory and transformed input $o = O(I(\mathbf{x}, \mathbf{m}))$

4. Finally, the output representation is used to decode the final response: $r = R(o)$

The components I, G, O, and R can be represented with any function capable of doing the task. However, in most cases in the literature, we see models that use neural networks to instantiate these components. Consequently, the whole model is end-to-end differentiable and thus trainable with algorithms such as backpropagation. In that case we talk about Memory Neural Networks (MemNNs). Although various kinds of data can be input and output to MemNNs in general, we consider only MemNNs that work with text in this work. Let us describe a basic MemNN for text. When given text input, such a basic model can save each text memory into a new memory slot. Therefore, old memories are not updated, and new inputs are stored sequentially. The most interesting components of this simple text model are O and R. The O module scores the saved memory vectors given text input x and subsequently retrieves k supporting entries with the highest scores from the memory. The R module takes the input x and k retrieved memories to produce a response. It generates an output by copying one of the memories or creating new outputs, for example, with an auto-regressive RNN decoder.

2.4.2 Multi-hop end-to-end MemNN

The MemNN architecture's disadvantage is that it requires supervision at each layer during training, as pointed out by Sukhbaatar et al. (2015). In this work, the authors address this problem and present a fully end-to-end trainable memory network model, which we call *e2e MemNN*. In the e2e MemNN, we input a set of vectors x to be stored in the memory and a query q. Each input x_i is then embedded with two distinct embedding functions E_m and E_c to obtain a memory entry m_i and corresponding output embedding c_i, respectively. The query q is then embedded with the different function E_q to obtain query embedding u.

Upon embedding the inputs, a memory distribution p is computed as a match between u and each memory m_i as $\text{softmax}(u^T m_i)$. The memory distribution p is then used to create an output o computed as a weighted sum over the output embeddings $\sum_i p_i c_i$. The output o is then used to produce the final answer, a categorical label or textual response generated with an auto-regressive model.

The authors also introduce a multi-layer version of this model. In the multi-layer version, each layer k has distinct embedding matrices E_m^k and E_c^k. The output o_k from layer k is used to form input query embedding u_{k+1} to the layer $k+1$: $u_{k+1} = u_k + o_k$. The multi-layer processing is sometimes referred to as *multi-hop* where a *hop* refers to processing by one layer.

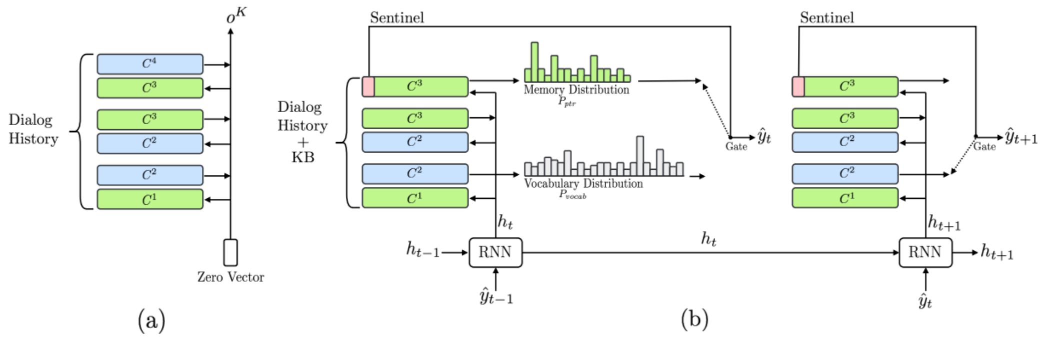


Figure 2.6: The Mem2Seq model (Madotto et al., 2018) showing the memory encoder with 3 hops (a) and 2 steps of the memory decoder (b)

2.4.3 Mem2Seq model

The Mem2Seq model (Madotto et al., 2018) is a task-oriented dialogue model that builds on top of the end-to-end architecture introduced in Sukhbaatar et al. (2015). The architecture is described in Figure 2.6. The model consists of encoder and decoder parts. The encoder is a 3-hop memory network that encodes the dialogue context. The more interesting part is the decoder, a recurrent neural network enhanced with a memory network in every time step. This memory network contains conversation history and knowledge base entries encoded as memory vectors. At each step, the hidden state of the RNN is used as a query vector. Then, a distribution over memory is computed, as described earlier. Also, a distribution over vocabulary is computed from the RNN hidden state, similar to standard RNN-based decoders. One special entry is added to the memory, so-called *sentinel*. The vocabulary distribution generates the next token if the highest probability is assigned to the sentinel token. Otherwise, the next token is chosen according to the memory distribution. In other words, the model uses the RNN state at each generation step to decide whether to copy something from memory (conversation history + knowledge base info) or generate an arbitrary new token from vocabulary distribution. This way, the model can use information from the past effectively.

USER:	*I would like a cheap restaurant.*	**inform (price = cheap)**
SYSTEM:	*Golden plate is cheap.*	**inform (name = Golden plate)**
USER:	*What is the cuisine?*	**request (cuisine)**
SYSTEM:	*They serve chinese food.*	**inform (cuisine = chinese)**
USER:	*Sounds good. Bye!*	**goodbye ()**
SYSTEM:	*Have a great day.*	**goodbye ()**

Table 2.1: Example of task-oriented dialogue in the restaurant reservation domain. Utterance representations as dialogue acts are depicted on the right. Intents are highlighted in orange, slot names in blue, and respective values in green. Note that not all dialogue acts include slots and values.

2.5 Dialogue System implementations

First, we describe various approaches to constructing dialogue systems pipelines and provide insights about approaches to modeling the underlying processes. The architectures may vary greatly because of the varying use cases of Dialogue Systems (DS). We thus introduce a classification of dialogue systems that reflects the expected capabilities. There are multiple approaches to defining a dialogue system taxonomy in the literature. Here, we introduce the widely used classification scheme (Jurafsky, 2000).

1. **Question Answering (QA)** - Although sometimes not mentioned in the context of dialogue systems, the QA task can be seen as a simple conversation. A QA system's main task is to answer the user's questions. The topics may vary; good understanding and knowledge representation are essential for this task. The dialogues are usually quite simple and often consist of just one question and the respective answer.

2. **Task-oriented DS** - In this setting, the system's goal is to complete a task based on the user's instructions. The successful completion may depend on several attributes the system has to learn from the user utterances. The system is also allowed to ask for additional information if needed and typically works with some external source of information, such as a database. Here, the dialogues are usually much more complex than in the QA setting, and dialogue context has to be considered.

3. **Chit-chat** - Sometimes, we might be interested in a system that can talk to the user casually and provide entertainment. Such systems might be combined with task-oriented systems to serve as human-like virtual assistants or use the dialogue to advertise products, etc. The context and knowledge base are also important, but in most cases, there is no well-defined task to be completed, so the evaluation is subjective.

Another way of classifying the dialogues considers their domain of operation. **Single-domain** systems can work only in one topic area, e.g. public transport or restaurant information, whereas **multi-domain** systems can handle multiple domains. These types of systems cannot give meaningful answers outside the domains they are trained on. A dialogue system is considered **open-domain** if it can have a conversation not limited to a predefined set of domains. In practice, this is achievable only to some extent since the knowledge base of the program is always limited. However, the systems can cover many domains with internet access and smart information retrieval methods.

Here, we focus on the task-oriented DS and discuss it more deeply. From the domain perspective, task-oriented DS are usually single or multi-domain systems. Open-domain dialogue systems are rare, at least in the research community. We can see a task-oriented dialogue as a composition of two tasks - slot-filling and response generation. Task-oriented dialogue systems are widely used for various applications, such as customer service, personal assistance, and information retrieval. These systems aim to assist users in accomplishing specific tasks by engaging in a natural language conversation. That means we have a predefined set of semantic *slots* that must be filled with the right *values*. Each utterance in the task-oriented dialogue is considered an action that potentially changes the state of the conversation. Such actions can be represented using *Dialogue Acts (DA)* (Core and Allen, 1997). To define the Dialogue Act, we first must introduce the concept of *slots* and *intents*. To represent the meaning of user utterances, annotation based on *slots* is commonly employed (Young et al., 2013). Slots, which describe semantic concepts relevant to completing the task, serve as a means of capturing the user's desires as well as facilitating the system's communication with the user. Typical examples of slots include *area*, *price*, and *address*, among others. By tracking slots and their values throughout the dialogue, a dialogue system can maintain a *dialogue state*, effectively planning the next actions (Williams et al., 2013). The dialogue state represents explicitly all the important knowledge known to the system at a specific point in the dialogue. Consequently, it can be utilized to communicate with external sources of information and data, such as databases, structured knowledge bases, or various APIs, to provide users with accurate and relevant information. Intent describes the user's intention expressed by a respective utterance. In other words, intent represents what the user wants and what their wish is. Slots describe attributes of this wish and ground it to an ontology.

DA is a tuple consisting of an *intent* and *slot* and the corresponding *value*. If multiple slot values are present, all are considered to have the same intent.

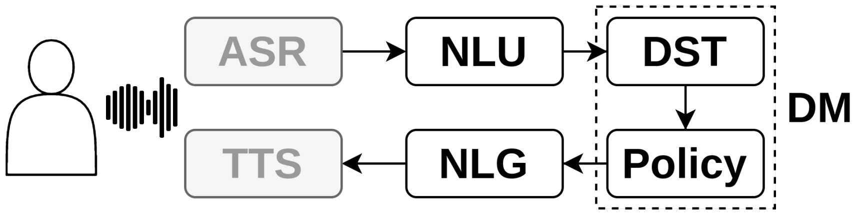


Figure 2.7: Overall architecture of task-oriented dialogue system pipeline. The data flow is outlined with arrows. This work does not discuss ASR and TTS modules (depicted in gray), often included with the rest of the components.

An example dialogue with respective DA representation is depicted in Table 2.1. Most dialogue system modules for limited domains can be implemented by designing a set of rules and templates. Such systems can yield satisfying results in some use cases. Nevertheless, they are inflexible and generally not considered promising from the research point of view.

2.5.1 Dialogue System Architectures

A typical approach to task-oriented DS implementation is to create a modular system with several modules that handle the conversation flow together. An example of such architecture is depicted in Figure 2.7 We shortly discuss the responsibilities of each component:

- **Natural Language Understanding (NLU)** The purpose of NLU is to extract the meaning of input utterances in natural language and transform it into a structured representation, i.e., dialogue acts. Basically, the NLU module has three subtasks. (1) It has to determine the domain of the utterance, (2) detect the user intent, and (3) capture any slot values, if present.
- **Dialogue State Tracking (DST)** Dialogue state keeps track of the dialogue history, effectively providing the necessary context. Dialogue State Trackers update the state with correct values after each turn.
- **Dialogue Policy** The core component of the DS is the dialogue policy. Its responsibility is to decide which action the system should take at each turn. In other words, Dialogue Policy's responsibility is to guide the dialogue to follow the desired path. Dialogue policy and the DST component are sometimes called *Dialogue Manager*.

- **Natural Language Generation (NLG)** When the decision on a system action is made, the system needs to verbalize the action. In other words, we need to create an utterance in natural language that expresses the information in the system's underlying representation.

The module-based approach is advantageous thanks to its good level of explainability. In case of low performance, we can track the respective modules' outputs and find the source of problems. On the other hand, error accumulation makes it difficult to recover from errors made by the modules early on in the pipeline. Another disadvantage is how these systems are trained (Li et al., 2017). Each component requires specific data annotation. Thus, obtaining a dataset suitable for training all components can be difficult and costly. Also, the system design itself is more complicated since it requires the implementation of multiple models. Because of the drawbacks, many current research works focus on dialogue modeling end-to-end, i.e., using systems without explicit components (see Section 2.5.3) that can be trained jointly.

One commonly used concept in end-to-end dialogue modeling is that of *delexicalization* (Wen et al., 2015a). When using delexicalization, the slot values in the text of the utterances are replaced by placeholders. This addresses the problem of data sparsity because the model can learn just the important concepts and does not have to remember full ontology or use the correct values. Also, delexicalized values are better suited for interaction with external interfaces – we can fill (lexicalize) them later in a deterministic way to prevent the model from hallucinating incorrect values. However, delexicalization also suffers from several drawbacks. First, it is not straightforward to perform and requires ontology knowledge. Second, the inverted lexicalization process is sometimes difficult to implement and might require complex rules.

2.5.2 Modeling TOD without supervision

Here, we introduce some challenges concerning TOD modeling without external structured supervision. We separately discuss selected components in the overall pipeline (Section 2.5).

Natural Language Understanding (NLU) Recall that the purpose of the NLU component in a traditional dialogue pipeline is to extract the meaning from input utterances in natural language and represent it in a structured way. Doing this unsupervised is challenging because we do not have the labeled data for training and do not know the structure itself in the first place. However, there are some possible ways to approach this problem. Works focus on automated annotation schema induction (3.3). Another way is to let the model handle NLU implicitly, i.e., effectively omitting the NLU step.

Action Selection One of the tasks of the dialogue manager is to select the next system action in a given situation. Most implemented systems implicitly perform this task, i.e., there is no explicit representation of the system action, just a surface realization (verbalized utterance). The reason for this option is that it requires fewer supervised labels, which might be hard and costly to obtain. However, modeling the information about system action can benefit the overall performance (Liang et al., 2020). We explore the possibility of implicitly learning this information by introducing a network bottleneck in Chapter 5.

Handling external interfaces Task-oriented dialogue systems must provide accurate and complete information based on user requests, which requires interaction with external interfaces such as databases or some structured knowledge base. To communicate with such external entities, we typically need to design some representation with a predetermined structure so we can construct API queries, etc. This can be troublesome in a setting where no supervision in the form of labels is present. Without a known structure, it is very challenging to design the communication protocol with external sources of information.

2.5.3 End-to-end dialogue modeling with Language Models (LMs)

The pre-trained LMs can generate fluent, natural-sounding text and work with structured data. It is desirable to leverage these properties for end-to-end dialogue modeling. However, their application is not straightforward. Task-oriented dialogues require interaction with external interfaces, so there is a need to obtain an intermediate structured representation to allow this interaction. A common approach (depicted in Figure 2.8), pioneered in (Lei et al., 2018b), proposes to split the generation into two steps: (1) belief state generation and (2) response generation. In the first step, we decode a structured representation that can be used to interact with external interfaces and extend the context. This representation must follow the exact structure to be easily parsable with a deterministic function that constructs the query. Once generated, we can parse it, construct

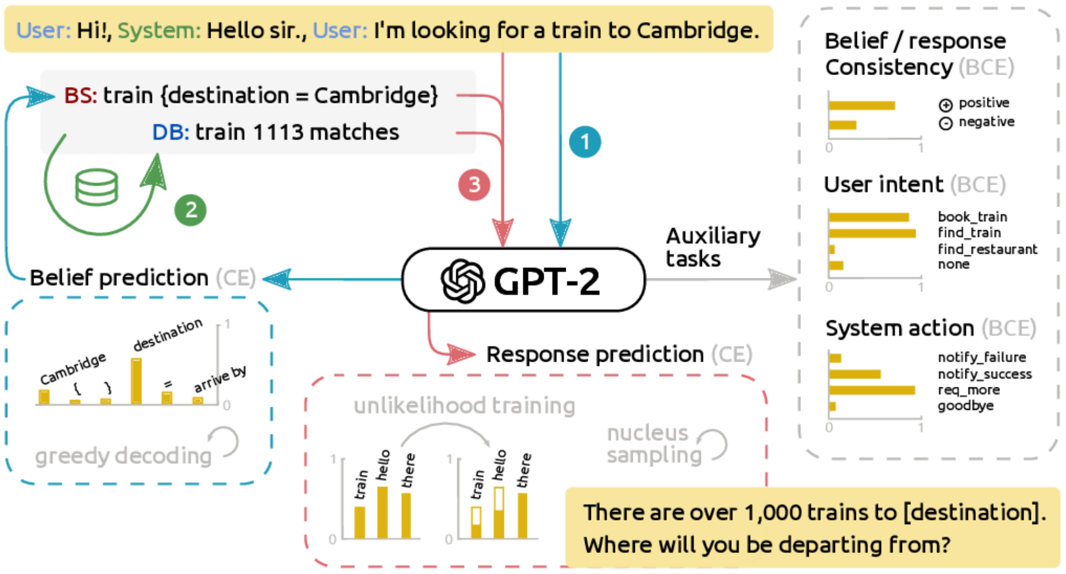


Figure 2.8: An explanation of two-stage LM-based dialogue model as described in (Kulhánek et al., 2021)

the query, and retrieve results. In the subsequent step (2), the model has all the needed information in the input and can generate the final response. Here, we usually generate delexicalized (Section 2.5.1) utterances that are later post-processed to fill in values from the retrieved results or generated state.

To train such an architecture, we can simply train the model jointly on belief state generation and response generation, with appropriate loss masking if needed. The database results are usually injected directly into the training data. During inference, the model is conditioned only on the left context and, therefore, can be applied for the 2-step generation process described above.

2.6 Datasets

We describe some of the most prominent datasets we use for our experiments. Most of them are largely used in the dialogue community and provide common benchmarks used for the evaluation. All these datasets have several task-oriented dialogue characteristics that define the conversation according to (Young et al., 2013):

1. **Domain(s)** define the topic (or range of topics) which are mentioned in the dialogue. There can be several domains per dialogue.

2. **Task**: the users in each dialogue attempt to reach a certain goal (such as booking a restaurant or finding a tourist attraction).

User	`I am looking for a Chinese restaurant`
System	`The Golden Dragon is a nice place in the South`
User	`Please, give me their phone number and address`

Table 2.2: An illustration of the difference between **inform** and **request** slots.

3. **Turns** We consider *turn-taking* dialogues, i.e. the participating sides exchange utterances alternately. One such utterance exchange is called a dialogue turn. Utterance is considered the linguistic realization of the speaker's thoughts and will. It can be spoken or written.

   The meaning of each utterance can be represented in a structured way with **Dialog Act(s)** (Weisser, 2016). It is a meta-information that emerges from the respective utterance and qualifies it. It describes the beliefs, desires, and intentions. Dialogue acts can be represented using *Domains*, *Intents*, and *Slots* as described in Section 2.5. The slots can also be further categorized. Typically, we define *inform* and *request* slots (Table 2.2). Inform slots correspond to information provided by the user and inform about constraints the user has, while request slots communicate the kind of information that the user wants to obtain. The intent represents the user's intention, i.e., the sub-goal that the user wants to achieve with a particular utterance. For an example of dialogue act annotation, we refer to Figure 2.1 in Section 2.5. Slots represent the attributes that instantiate the dialogue act. Each domain is associated with certain intents; each intent can be combined with multiple slots. A slot, however, can be used by multiple intents as well. We provide statistics about the dialogue datasets we use in this work in Table 2.3 and some samples in Tables 2.4 and 2.5.

2.6.1 Datasets description

MultiWOZ (**MW**) is an established task-oriented dataset introduced by Budzianowski et al. (2018a). It has been released in several versions; the standard most commonly used nowadays are MultiWOZ 2.1 and MultiWOZ 2.2. MultiWOZ 2.2 is a version of the original dataset improved by (1) fixing some annotation errors, inconsistencies, and ontology issues and (2) adding slot span annotations for utterances. MultiWOZ contains over 10,000 annotated dialogues and spans multiple domains – restaurant and hotel reservations, tourist attraction search, and taxi and train reservations. While some of the dialogues use only a single domain, most of them are multi-domain. For example, after finding a restaurant, the user asks for a hotel and orders a taxi. This makes the dataset more relevant to real-world use cases.

The data were gathered via a crowd-sourcing Wizard-of-Oz scheme described in Wen et al. (2017c). As MW is used most in our work, we provide further information on the data distribution in Figure 2.9 and a sample of the data in Table 2.4.

DSTC2 (Henderson et al., 2014) was introduced as a part of a challenge to improve state tracking within dialogue systems. It contains over 3,000 dialogues covering a single domain around restaurant reservations. The dialogue corpus was collected using Amazon Mechanical Turk[1] with a POMDP-based spoken dialogue system. It is the only human-machine dataset in our collection.

CamRest676 (**CR**) (Wen et al., 2017d) is another crowd-sourced dialogue corpus gathered via the Wizard-of-Oz scheme. CamRest676, with its 676 conversations, is the smallest of the datasets used in this work, and it is also a single-domain dataset focused on helping users to find a restaurant in Cambridge, UK.

Schema-guided dialogue (**SGD**) is a large (more than 20,000 dialogues) multi-domain (around 20 domains covered) dataset containing a total of 45 API services based on a pre-defined schema. First, the data was collected via a simulator that interacts with the API services, and then the dialogues were paraphrased using crowd-sourcing.

ATIS (**AT**) (Hemphill et al., 1990) contains utterances taken from conversations about flight searches and reservations. [2]

Cambridge SLU (CS) (Henderson et al., 2012) resembles the CamRest 676 dataset but is larger and focuses only on other user parts of the conversations. Therefore, Cambridge SLU is not a true dialogue dataset as it contains only single utterances and can be used solely for the NLU task.

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

[2]There are multiple ATIS data versions available. We used one from `https://www.kaggle.com/siddhadev/atis-dataset-from-ms-cntk`.

Stanford Multidomain Dialogues (**SMD**) (Eric et al., 2017) contains concise dialogues between a driver and an in-car virtual assistant about appointments, navigation, and weather. The dataset assumes interaction with the database or external APIs. However, the information is provided with each dialogue in the form of relevant Knowledge Base entries. We provide an example from the data in Table 2.5.

2.6.2 Dataset limitations

Although the number of dialogue datasets is quite big, and there is arguably a lot of variance with respect to the size, data collection approach, etc. there are some limitations that are inherently present with this type of data. We want the data to be used for two main purposes: training and evaluation. Both these aspects are influenced by the static nature of the data. Dialogue is a dynamic process that requires interaction. Hundreds of alternatives exist for a conversation representing a certain information exchange with different phrasing, length, or turn ordering. All of this is impossible to capture in a static dataset. Therefore, dialogue modeling should somehow consider this. A similar issue stems from the fact that there are multiple valid responses under a specific dialogue context, not only with respect to the phrasing but also considering dialogue acts. Consequently, model evaluation with these data tends to penalize generated responses that are meaningful and relevant but do not correspond to the ground truth found in the data.

Another consideration regards the focus of these datasets. We almost exclusively work with *task oriented* datasets in this work. Therefore, the instances contain some structured annotation, most frequently in a dialogue state. While this addresses the above-mentioned issues to some extent, the conversations usually lack variability and do not exhibit user behavior like repetitions, small talk, clarifications, hesitation, etc. Therefore, there is a substantial gap with respect to real-world data distribution. On the other hand, the other type of dialogue datasets tries to exhibit these properties but lacks any structure. Recently, some efforts were made to join these two approaches (Sun et al., 2021).

Data	SGD	MW	DSTC	CR	SMD	ATIS	Total
Domains	18	7	1	1	3	1	19*
Slots	145	29	10	7	15	79	166*
Dialogues**	22.8	10.4	3.2	0.7	3	–	37.1
Turns**	463.3	143.0	51.0	5.5	16.1	4.9	662.8
Turns/Dial.	20.30	13.71	15.77	8.12	5.25	–	17.83
Avg. utt. length	9.86	13.23	8.47	10.71	9	11.37	10.49
Unique Words**	32.3	23.2	1.3	1.7	1.6	0.9	49.9

Table 2.3: Basic statistics of the datasets we use in this work. Overall and for individual sources (number of domains and slots, total numbers of dialogues and turns, average number of turns per dialogue, and average utterance length in terms of words. Due to ontology overlap, * is not a sum. ** in thousands.

User	`I need to book a hotel in the east that has 4 stars.`
System	`I can help you with that. What is your price range?`
State	`restaurant {}, ..., hotel {"area": "east", "stars": "4" }`
User	`That does not matter as long as it has wifi and parking.`
System	`If you'd like something cheap, I recommend the Allenbell.`
State	`restaurant {}, ..., hotel {"area": "east", "stars": "4", "wifi": yes", "parking": "yes"}`
	`...`

Table 2.4: A simplified example taken from the MultiWOZ corpus. It shows a snippet from a conversation between the customer and the agent about booking a hotel. It also shows this corpus' annotation schema for tracking belief state.

Driver	`What gas stations are here?`
NLU	`{"poi_type": "gas stations" }`
Car	`There's a Chevron`
Driver	`That's good! Please pick the quickest route to get there and avoid all heavy traffic!`
NLU	`{"distance": "quickest", "traffic_info": "avoid all heavy traffic"}`
Car	`Taking you to Chevron`
KB	`"items": [ {"distance": "5 miles", "traffic_info": "moderate traffic", "poi_type": "gas station", "address": "783 Arcadia Pl", "poi": "Chevron"} ... ]`

Table 2.5: A simplified example taken from the SMD corpus with utterance-level annotations. Compared to MultiWOZ, some of the slot values are open, such as ”avoid all heavy traffic”

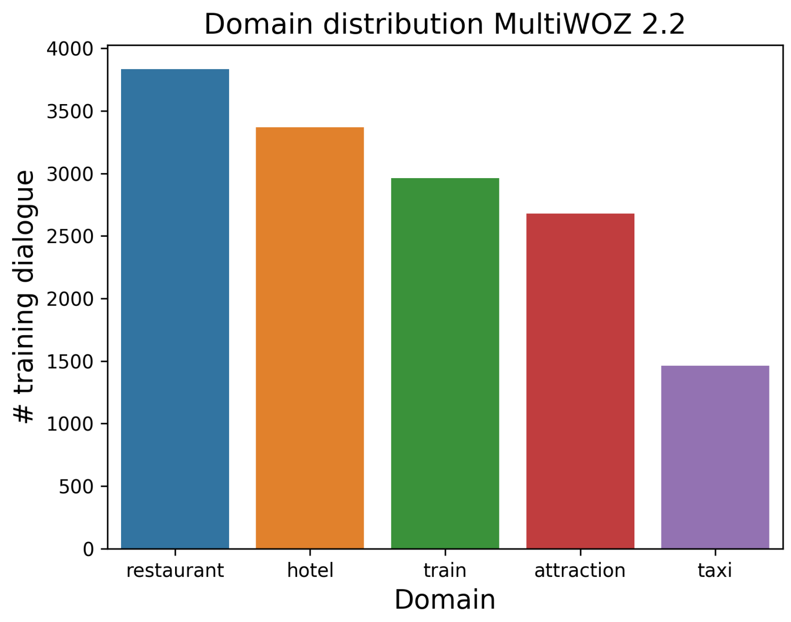


Figure 2.9: The distribution of respective domains in the MultiWOZ dialogue dataset, showing how many of the dialogues contain the respective domain.

2.7 Evaluation metrics

Here, we provide a set of commonly used metrics to evaluate the quality of dialogue modeling and response generation. We also use some less common metrics for specific tasks. Those metrics are described together with experiments in respective sections of this work.

2.7.1 NLU metrics

- F_1 **Score** (F-score, F-measure) (Goutte and Gaussier, 2005) is a widely used metric to evaluate binary classification. To measure F1 score, we first compute the number of occurrences of True Positives (TP), False Positives (FP), and False Negatives (FN). Subsequently, we can compute Precision (P) and Recall (R) in the following way:

$$P = \frac{TP}{TP + FP}; R = \frac{TP}{TP + FN}$$

The F_1 score is then computed as a harmonic mean of P and R:

$$F_1 = \frac{2 \cdot P \cdot R}{P + R}$$

F_1 score is sometimes used also to measure multiclass classification performance. Usually, two ways of F_1 generalization are used. *Micro F_1 score* averages per-class F_1 scores with weights corresponding to the respective class frequencies whereas *Macro F_1 score* considers all classes equally important. F_1 is a classification evaluation metric task, and in the context of dialogue NLU it is frequently used to evaluate the performance of slot filling.

- **Intent Accuracy** is the percentage of slot occurrences assigned into the correct intent cluster under the reference mapping.
- **Domain Detection Accuracy** is simply the ratio of cases in which the system correctly detects the domain.
- **Entity Match Rate (EMR)** (Wen et al., 2017d) calculates the last turn's entity in each dialogue. Using the final constraints, it verifies if a correct entity would be retrieved from the database.

2.7.2 State tracking metrics

- **Joint Goal Accuracy (JGA)** (Mrkšić et al., 2017) is computed as the ratio of dialogue turns for which the predicted belief state matches the ground truth. We use fuzzy matching of the slot values so that capitalization or minor typos do not influence the result.
- F_1 score on the slot level can also be used to evaluate the performance of state tracking.

2.7.3 Dialogue-level metrics

- The main *overall measure* for evaluating a task-oriented dialogue is the dialogue **success rate** (Deriu et al., 2021). For MultiWOZ, we use the standard evaluation of dialogue success as the ratio of dialogues where the user reaches the desired goal, based on goal annotation provided with the data (Nekvinda and Dušek, 2021a). The SGD dataset does not include goal annotation but contains information about the requested slots. Therefore, we compute SGD success rate as the proportion of dialogues in which (1) the system captures all the informed slots correctly and (2) all the requested slots are provided. For more details about the inform and request slots, we refer the reader to Section 2.6.

- **BLEU score** (Papineni et al., 2002) is largely used to evaluate response generation in machine translation, summarization, and other tasks, including dialogue generation. Although it has some flaws (Callison-Burch et al., 2006) and was designed mainly for corpus-based evaluation, it is widely used, so it is desirable to measure it for comparison with other works.

There are multiple different criteria for dialogue systems evaluation. In the case of modular systems, the individual modules can be evaluated separately. It is considerably harder to measure a system's overall performance. The policy module's decision is difficult to evaluate without turn-level action annotations. BLEU usage is controversial for dialogue systems since it often fails to capture the semantics of the utterance, which is perhaps more critical than in the translation task (Lowe et al., 2017). It is also common to measure **Dialogue success rate**. However, it is not straightforward how to define dialogue success robustly. Many systems use user simulators to allow the employment of reinforcement learning techniques. In such scenarios, the user behavior is model-based and can be non-deterministic. Therefore, defining success is challenging. Usually, it is based on evaluating user and system dialogue acts, which rely on good and extensive data annotation.

2.7.4 Human Evaluation

Due to the mentioned challenges, **human evaluation** remains the best way to evaluate dialogue systems. Human evaluation plays a crucial role in developing and refining dialogue systems. Automated tools or metrics often fail to capture human communication's nuanced responses and language variations.

Dialogue systems, including chatbots and virtual assistants, are directly designed for human interaction, making their performances depend heavily on how effectively they can understand and respond. Therefore, assessing their capabilities requires a human-like understanding of the conversations.

Moreover, human evaluation is important to measure aspects beyond the correctness of the system's responses. For instance, the engagingness of the conversation or the appropriateness of the system's tone requires human judgment. Also, human evaluators can better identify and evaluate cultural and social aspects than automated methods.

While human evaluations are more time-intensive and costlier than automated ones, they offer a more detailed, insightful, and accurate assessment of the dialogue system's performance. Therefore, they are crucial for improving these systems to make them more user-friendly and effective for human interaction.

3 Related Work

Here, we introduce some of the less-known models and approaches that address the same tasks we are trying to solve. The purpose of this chapter is to put our work in context, offer a comparison, and provide a broader intuition about what our ideas are based on. In Section 3.1, we introduce traditional approaches to task-oriented dialogue modeling, implementing components for natural language understanding, dialogue state tracking, or dialogue policy. We follow with end-to-end approaches in Section 3.2. Finally, we describe various methods focusing on unsupervised approaches and transfer learning in Section 3.3.

3.1 Modular dialogue architectures

The traditional dialogue system implementation, especially for task-oriented dialogues, is based on modular architecture. The modular system consists of several components connected to form a pipeline. First, the Natural Language Understanding (NLU) module parses the utterance and creates a structured representation. Based on NLU outputs, the dialogue management module determines the next action. Dialogue Management usually consists of the state tracker that updates the state based on NLU outputs and the policy module that chooses the action. Finally, the language generation module is used to verbalize the chosen action.

3.1.1 Natural Language Understanding (NLU)

From the machine learning point of view, the intent and domain detection can be seen as a classification task, and sentence-level classification can be utilized (Yaman et al., 2008; Schapire and Singer, 2000). The slot-value filling can be approached as a sequence tagging problem. Many approaches have been proposed to tackle this issue, ranging from SVM (Shi et al., 2016) and HMM (Surendran and Levow, 2006) based taggers to various neural models (Adel et al., 2016; Zhang et al., 2017; Mesnil et al., 2014). It is reasonable to model them jointly because of the similar nature of these three sub-tasks. Modeling the intent detection together with slot filling proved to be beneficial for the model performance (Zhang et al., 2017; Liu and Lane, 2016; Xu and Sarikaya, 2013). Slightly different approach was introduces in Liang et al. (2020) who introduce end-to-end trainable systems that receives module-level supervision and hence can be seen as a modular architecture. Some of the newer approaches allow to switch between tasks with tree-based system (Xie et al., 2022).

3.1.2 Dialogue State Tracking (DST)

The most straightforward solution to this problem is a rule-based system that tracks the current slot values based on NLU. However, the situation is usually more complicated. We must consider a distribution of slot value probabilities, and the update rules can be rather complex. Žilka et al. (2013) compares different data-driven models for dialogue state tracking. Neural networks have also been used to model the distributions (Mrkšić et al., 2017; Zhong et al., 2018) and deal with multiple domain handling (Rastogi et al., 2017). Some recent works explore the usage of language models for dialogue state tracking (Lee et al., 2021; Hu et al., 2022b).

3.1.3 Dialogue Policy

The policy decision can thus be framed as a classification task (Gašić and Young, 2013). Learning the policy just from the offline data might not produce robust policy due to low variability in the data. Therefore, many works model the dialogue as a partially observable Markov decision process (Gašić et al., 2010; Thomson and Young, 2010). Reinforcement learning techniques are then applied to learn the policy and incorporate human feedback (Peng et al., 2017; Su et al., 2016). Some approaches also try to model the actions in latent space and tune them via RL (Lubis et al., 2022). Again, new line of works uses language models and prompting methods such as Zhang et al. (2023).

3.1.4 Natural Language Generation (NLG)

NLG is often realized with a set of handcrafted templates that are selected heuristically (Rudnicky et al., 1999). The variability of the generated utterances is limited, and the scalability is poor. Therefore, corpus-based methods have been proposed (Oh and Rudnicky, 2000; Mairesse and Young, 2014). Neural network based systems were proposed as well (Wen et al., 2015b, 2016). Another approach combines schemas and templates (Kale and Rastogi, 2020) or use more sophisticated semantic representations (Balakrishnan et al., 2019)

3.2 End-to-end architectures

Various end-to-end solutions have been proposed to address the drawbacks of modular system training (discussed in Section 2.5). This was made possible largely thanks to the growing popularity of Neural Networks (NN) and the backpropagation algorithm over the last decade (see Section 2.1.1). NN forms a family of models that naturally allow us to combine and train multiple models using a single training algorithm. Therefore, several solutions were proposed to implement the respective modules using neural network-based models, interconnecting them to form the pipeline and train them jointly (Li et al., 2017; Wen et al., 2017b). Although the end-to-end training improves the scalability of the models, the proposed architectures still require multiple levels of data annotation for training. To mitigate this problem, Serban et al. (2016) proposed a hierarchical end-to-end model that uses two levels of encoder-decoder Recurrent Neural Networks (RNN), one operating on dialogue turn level for keeping long-term context and one operating on word level for analyzing the current user input. It does not follow the traditional pipeline scheme and thus does not require expert annotations. However, it is unsuitable for practical use in task-oriented DS in its raw form due to its low performance and insufficient robustness. The idea was further extended by Williams et al. (2017b), who introduced the *Hybrid Code Networks*, an architecture that uses multiple utterance representations customizable by the developer. Despite good performance and flexibility, the proposed model again required a non-trivial amount of data annotation. Lei et al. (2018b) devised a novel idea to model the dialogue with an extended sequence-to-sequence model. They use an encoder-decoder architecture based on RNN that generates a dialogue state before response generation. They summarize the dialogue history in the RNN hidden state and use a system of copy mechanisms to track the dialogue state. The proposed dialogue state representation is greatly simplified and does not require explicit NLU input. Thus, the annotation process is significantly easier.

In recent years, the NLP world has witnessed the great success of attention-based models (Transformers) (Vaswani et al., 2017) and their usage as pre-trained language models (Devlin et al., 2019a). In dialogue systems, these models also show prominent results in the open-domain setting (Wolf et al., 2019) or for dialogue state tracking (Chao and Lane, 2019). The pre-trained models are naturally utilizable for transfer learning, which proved useful in dialogue domain adaptation task (Shalyminov et al., 2019b). Recently, attention-based architecture was proposed that models latent dialogue actions (Bao et al., 2019).

Task-oriented dialogue modeling with the use of pre-trained language models was researched by Zhang et al. (2019) or Peng et al. (2021a), who followed the ideas of text-based state encoding and 2-stage generation proposed in the Sequicity model (Lei et al., 2018b). This approach first uses a text-based model to decode the structured belief state. The belief state is later used to retrieve db information optionally, and finally, the model is called once more, conditioned on the belief state and retrieved information to generate a response. Several other improvements were proposed to the architecture that either improve contrastive state training (Kulhánek et al., 2021) or redefine state tracking as generation of belief state differences (Lin et al., 2020). Others also proposed combining the purely generative models and retrieval-based approaches (Pandey et al., 2018; Cai et al., 2019; Nekvinda and Dušek, 2022). The above-mentioned works fine-tuned the model on the in-domain data, contrasting with the pure in-context learning approach we apply in Chapter 7.

3.2.1 Instruction Tuning for Large Language Models

Here, we follow up on the introduction to Large Language Models (LLMs) in Section 2.2. In particular, we discuss instruction-tuning techniques that aim to make LLMs more accessible. The idea of using reinforcement learning techniques to align model-based agents better with users' intents was pioneered in game agent development. (Christiano et al., 2017) and later explored for training language models (Ziegler et al., 2019; Ouyang et al., 2022). Although these techniques proved quite effective, the process is still very demanding in collecting user feedback. Consequently, several datasets were proposed (Wang et al., 2022; Iyer et al., 2022; Black et al., 2022) that collected millions of instructions-based tasks in natural language and can be applied to align LMs similarly to reinforcement learning.

3.3 Unsupervised and transfer learning methods

The research of methods that reduce the amount of supervision needed can be divided into two paradigms. One research direction tries to construct a method of unsupervised or weakly supervised data analysis, focusing on a certain part of the dialogue pipeline. Such a method can provide artificial supervision for the supervised models introduced earlier. The other option is to design a model that inherently does not need supervision or requires less annotation.

3.3.1 Unsupervised analysis and labeling for NLU

Various methods have been proposed to deal with NLU without explicit supervision. Chen et al. (2016) first proposed a model for zero-shot user intent embedding prediction by training a convolutional neural network to score the sentence-intent similarities. Recently, Shi et al. (2018) proposed an intent detection model using sentence clustering based on sentence-level features. They have applied their method successfully for the task of intent detection.

Using semantic relations to perform language understanding in the unsupervised setting was proposed by Heck and Hakkani-Tür (2012). Here, the authors use the Semantic Web (Berners-Lee et al., 2001), a triple-based entity relations database. Their approach relies heavily on structured web pages for the target domain. They exploit the structure to obtain semantic annotations in an unsupervised setting.

Chen et al. (2014) combine the paradigms of semantic frame parsing with distributional semantics to perform unsupervised semantic slot induction. The authors further improve their model in Chen et al. (2015) where they select the most prominent slot candidates using lexical knowledge graphs. However, both approaches only output a ranking of potential slot candidates based on frames. Since frame annotation is very fine-grained, this produces many candidates, requiring their manual merging into slots for any practical use. In contrast, we determine domain-relevant slots automatically. Coope et al. (2020) focus on a few-shot setting and perform span extraction of slot values using pre-trained models in Chapter 4. Another direction of research focuses on zero-shot slot filling. Bapna et al. (2017)'s recurrent-neural-network-based slot tagger is pre-trained on multiple domains and takes a textual description of the target slot on the input in addition to the user utterance. This way, adapting to a new domain only involves providing new slot descriptions. Further works extend this idea with more complex architectures (Shah et al., 2019; Liu et al., 2020). An interesting follow-up work was presented in Yu et al. (2022) which discovered dialogue schema slot candidates by analyzing attention spans of pre-trained LMs and clustering the spans with the DBSCAN algorithm. Recently, Qiu et al. (2022) proposed an alternative way to dialogue slot discovery by using pre-trained sequence tagging models based on BERT taggers.

3.3.2 Dialogue structure discovery

Brychcín and Král (2016) focused on modeling the dialogue as a Markov decision process using HMMs. By fitting the HMMs to the data, they explore the dialogue dynamics and assign Dialogue Acts to the HMM states. Later, people based the structure discovery on the VRNN-based models. In the DVRNN model (Shi et al., 2019), the authors train an unsupervised VRNN model with discrete latent states and explore transition probabilities of the neighboring latent states to explore the dialogue structure. This approach is later improved by Qiu et al. (2020), who propose to augment VRNN with the CRF layer in their SVRNN model. Yet another work that combines LM-based representations and HMM is presented in Lu et al. (2022).

3.3.3 Modeling dialogue generation with less supervision

Work regarding using semi-supervised or unsupervised methods for the dialogue response generation task as a whole in the task-oriented setting has been limited so far. One of the main challenges is to model the dialogue state with no supervision since it is structured and might be quite complex.

The method proposed by Jin et al. (2018) builds on Lei et al. (2018b)'s sequence-to-sequence dialogue model (see Section 3.2) by introducing a posterior regularization term in the loss function. The model has two modules, a teacher and a student, to track the dialogue state and works semi-supervised. For supervised data, both tracker modules are trained with supervised classification loss. For unsupervised data, the teacher module can look at system responses, therefore, it operates with more input information and makes more accurate predictions. The student module is then trained to minimize the KL divergence loss. The teacher module is conditioned on the system response, so it cannot be used when the model is deployed, but it helps to train the student even with unlabeled data.

Wen et al. (2017a) introduced a model that learns latent intentions, bypassing the explicit dialogue state modeling. Zhao and Eskenazi (2018a) approached the problem differently. They designed a novel dialogue system model based on VAEs. Their model uses supervised data from one domain to learn latent action representations. Their recognition module is learned to map utterance representations to the same feature space as the action representations. When transferring to another domain, the model needs only a few seed responses to adapt. Based on this idea, other works followed (Shalyminov et al., 2019b; Huang et al., 2019).

3.3.4 Few-shot dialogue modelling

One of the first neural network based models focusing on learning dialogue from a few in-domain examples was the Hybrid Code Networks (Williams et al., 2017a), a trainable system based on recurrent neural networks with partially handcrafted components. Another approach was proposed in Zhao and Eskenazi (2018a), which used latent action representations to transfer domain knowledge. Latent actions were also used in Huang et al. (2020) or Shalyminov et al. (2019a). More recent approaches use the Transformer architecture and pre-trained language models (Shalyminov et al., 2020) to leverage these models' abilities obtained during large-scale pre-training. Another example is Madotto et al. (2020) or Hu et al. (2022a), which used LLMs and in-context learning to perform belief state tracking. They did not use instruction-tuned models and formulated the task as an SQL query generation. However, they omit the response generation as well as database retrieval.

4 Discovering dialogue slots

Getting raw, unlabeled data for dialogue system training is not difficult, especially if we restrict the target domain. In general, recording conversations in real life or artificial conditions is sufficient. A requirement for dialogue state labels, which we discuss in Section 2.5, makes this process much more costly. The sets of slots and their values typically must be designed by domain experts. This procedure consists of multiple tasks:

1. Determine which concepts need to be captured.
2. Define the captured concepts in a consistent way.
3. Label the occurrences of these concepts in the training data.

As mentioned, these steps require expert knowledge and sometimes non-trivial domain understanding. While a dialogue system does not necessarily have to rely on the usage of slots, both traditional pipeline systems (Young et al., 2013) and end-to-end task-oriented architectures (Wen et al., 2017d) typically require such annotation. While some systems presented in Section 3.1 use implicit, latent state representation and do not require explicit labels, the behavior of such systems is hard to interpret or control, which can be crucial in practical applications. Moreover, slots enable communication with external interfaces, as discussed in Section 2.5. Several works are aiming at keeping interpretability and reducing the annotation needs by automating it (Chen et al., 2014, 2015) or transferring annotation across domains (Zhao and Eskenazi, 2018b; Coope et al., 2020), but they still require a significant manual effort. We present a novel approach to discovering a set of domain-relevant dialogue slots and their values given a set of dialogues in the target domain (such as transcripts from a call center). Our approach requires no manual annotation to tag slots in dialogue data. This substantially simplifies the dialogue system design and training process, as the developer no longer needs to design a set of slots and annotate their occurrences in the training data. We present the overview of our method in Section 4.1. Next, we go through the details of each stage. We discuss the experimental setup in Section 4.6 and the results in Section 4.8.

Most of the contents of this chapter were published at ACL 2021 (Hudeček et al., 2021). We also present extensions to the published content to remove the requirement for third-party weak-supervision annotation models and put our method in the context of instruction-tuned LLMs. These extensions are discussed in Section 4.4 and reflected in the following experimental sections. Our experimental code is available on GitHub[1].

4.1 Method overview

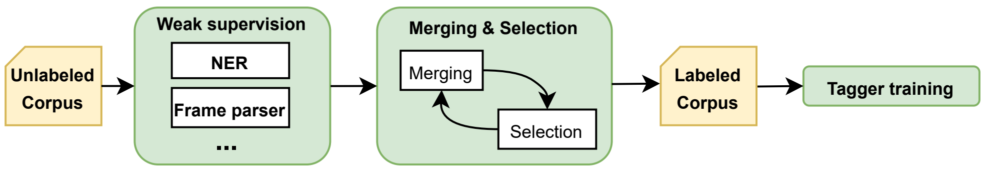


Figure 4.1: Illustration of our pipeline. First, we analyze an unlabeled in-domain corpus with supplied domain-agnostic linguistic annotation models, such as a frame-semantic parser or NER. This results in slot candidates. Next, we iteratively merge and select slot candidates to obtain domain-relevant slots. Finally, we use the resulting slot labels in the corpus to train a neural slot tagger.

Figure 4.1 depicts a diagram describing our approach. Our slot discovery method has three main stages:

1. We obtain weak supervision labels from automatic generic annotation. We obtain this annotation using domain-independent natural language taggers such as a semantic frame parser or a named entity recognizer (NER). These models can detect important and relevant concepts in natural language utterances and subsequently group them using a set of pre-determined generic labels. Nevertheless, the raw output of these models is not polished and cannot be used for the dialogue system directly. For more details, see Section 4.2.

[1]`https://github.com/vojtsek/joint-induction`

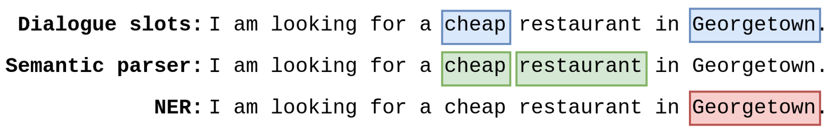
2. We identify domain-relevant slots based on the annotation labels by iteratively (a) merging and (b) ranking and selecting the most viable candidates (Section 4.4).

3. We use the discovered slots to train an independent slot tagger (Section 4.5).

We refer to this approach as ”weak supervision” because it uses noisy labels as input instead of the desired ones.

Consider an example in Figure 4.2. First, we note that it helps to use multiple tagging models since a respective model might not capture some of the concepts. The set of tagged words from all the sources covers all the dialogue slot values (*cheap*, *Georgetown*). However, it contains irrelevant words (*restaurant*). This behavior is expected since the tagging models are trained on generic open-domain data and detect all semantic concepts mentioned in the utterances.

Therefore, to exploit the output of generic models, we need to polish and customize it to the specific domain. Our method combines multiple sources of semantic labels and selects only relevant slot candidates. Slots discovered by our approach can then be used to design a schema relevant to a specific domain.



```
Dialogue slots: I am looking for a cheap restaurant in Georgetown.
Semantic parser: I am looking for a cheap restaurant in Georgetown.
            NER: I am looking for a cheap restaurant in Georgetown.
```

Figure 4.2: An utterance from the restaurant recommendation domain tagged with generic semantic parser (green) and Named Entity Recognition system (red). We provide a comparison with ground truth dialogue slot labels (blue).

4.2 Slot candidate identification by tagging semantic concepts

Our approach to selecting candidates for our method requires an initial pool of carefully chosen options representing coherent concepts. This step is critical to ensure the effectiveness of our selection process. We strive to gather as many candidates as possible to achieve this goal while preserving the above constraint. One of the key features of our method is its ability to merge several concepts into one, which means that we aim for high granularity and specificity in our input labels. As a result, we need to ensure that each candidate represents a unique, distinguishable concept.

Given that we cannot rely on human annotations, we use an automatic procedure to gather the initial set of candidates. This procedure combines multiple sequence tagging models to label the input corpus. This procedure aims to identify words or phrases in the text representing distinct concepts that can be used as candidate labels. We can use any sequence tagging NLP model that meets the following criteria: (1) a set of words with the same label indicates semantically coherent, distinct concepts, (2) no additional annotation is needed, and (3) the model is domain-independent.

For our experiments, we chose two types of taggers to obtain the input tags: Frame Semantic Parser and Named Entity Recognition (NER). By leveraging these models, we can quickly and accurately identify candidate labels that meet our criteria.

.

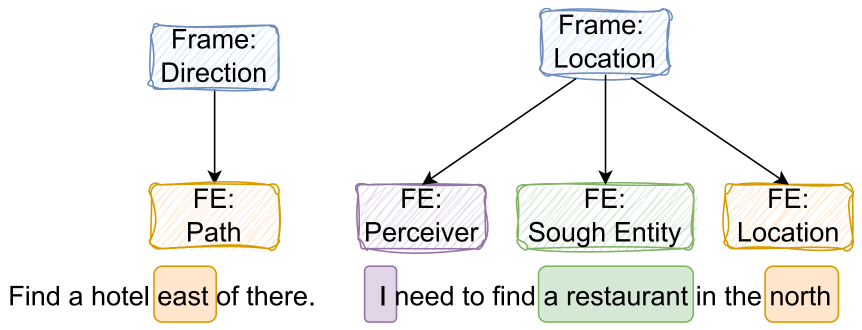


Figure 4.3: An example of two Frames defined in the FrameNet dataset, together with core Frame Elements and respective instances. In this example, we can see that a semantic concept representing the location of some place can be captured by multiple frames (*Direction* and *Location*). However, from the perspective of dialogue systems, these differences are negligible. Therefore, we merge some of the candidates to obtain a simpler schema.

Frame Semantic Parser The parser is based on the FrameNet project (Baker et al., 1998). FrameNet is a lexical database of the English language (although similar datasets exist in other languages) that aims to represent the usage of words in actual texts. It contains over 200,000 utterances with over 1,200 frames, each representing one semantic concept. From the NLP perspective, it can be approached as a task of Semantic Role Labeling. Each frame is formed by one or more frame elements, which together form an instance of a certain semantic situation (e.g., *Locale*, *Offenses*, *Size*,...). See Figure 4.3 for the example of FrameNet instances. We take the individual Frame Elements as a source for our slot candidates.

I	want	to	leave	from	Edinburgh	Waverley.
O	**O**	**O**	**O**	**O**	**B**	**I**

Table 4.1: An example of BIO tagging to identify named entities.

Named Entity Recognition (NER) is a well-established task of labeling occurrences of certain named entities in the input text data. Typically, the NER labels *word spans* that belong to a particular entity in the annotated utterance. A common approach to identify the spans is BIO tagging (Ramshaw and Marcus, 1995). See the example in Table 4.1. It fits our requirements well because the task is universal, and the named entities definition is not specific to any particular domain.

4.3 Unsupervised candidate identification

The approach introduced in this chapter is limited by the requirement of third-party models to gather the initial set of candidates. Therefore, we experiment with alternative ways of obtaining slot candidates completely unsupervised. This can lift the requirement of using generic models fine-tuned on out-of-distribution data.

4.3.1 Memory Networks for candidates identification

The end-to-end Memory Networks architecture described in Chapter 2.4 suits our needs for an unsupervised slot candidate identification. We train the Mem2Seq (see Sec. 2.4.3) model on the target dataset to learn the copy behavior. We then use the trained model to obtain a set of slot candidates. To achieve this, we let the model generate the test dialogues individually and inspect its behavior.

The Mem2Seq model's knowledge base entries represent entities or their properties, such as addresses. Furthermore, the model saves the conversation context via the memory mechanism. The content of the memory is further used to copy its entries directly to the produced response. Specifically, during the response generation process, the model decides whether the new token will be generated from decoder vocabulary distribution or copied from memory. The intuition is that when a certain token should appear in the generated response, the model decides to copy it rather than generate it to prevent mistakes. We save each token that is copied into the produced response rather than generated. These

Prompt	

	`Extract important concepts or objects`
	`from the following utterance.`
	`Yield all the concepts in one-level JSON dictionary.`
	`Include only entities and concepts that`
	`are mentioned in the utterance.`
	`Don't provide the intent.`
	`See the following examples:`
	`---------- Example ------`
	`Utterance: "I am looking for a theater`
	`that is reasonably priced."`
	`Output: {'venue': 'theater', 'price': 'reasonable'}`
	`-------------------------`
	`Now complete the following example:`
	`Utterance: "I am looking for a cheap hotel."`
	`Output:`

Table 4.2: The prompt, which is used to obtain slot candidates from the input utterance. It contains an example to specify the desired output structure

tokens will likely be slot candidates as they supposedly correspond to entities and slot values mentioned in the context. We then run k-means clustering to obtain a set of slot candidates. This way, we obtain multiple groups of word forms that can be used as input for our slot discovery pipeline.

We also note the similarity of this approach to the two-step generation process used in multiple architectures (Lei et al., 2018b; Peng et al., 2021b). In the first step of this process, the model generates placeholders such as *[address]* instead of specific values. This delexicalized utterance is then lexicalized in the second step by replacing the placeholders with actual values, e.g. using the database. This is almost exactly what happens in the Mem2Seq model. Only the two steps are joined in one, so the model puts the database values indirectly. Also, the lexicalization procedure of placing values instead of placeholders is learned rather than hardcoded.

4.3.2 LLMs for candidate identification

Another way of obtaining the slot candidates is by employing Large Language Models. These models can be instructed in the input text (prompted) to perform virtually any task (see Section 2.2). In our case, we instruct the model to extract the potential slot candidates directly from the input utterance. We use two models: *Tk-Instruct-11B* (Wang et al., 2022), which is T5 encoder-decoder architecture (Raffel et al., 2020a) tuned on a dataset of over 5M task instances with instructions and *ChatGPT* which is a closed source product introduced by the

OpenAI company[2]. The used prompt is given in Table 4.2. We use both models in parallel to obtain two sets of candidates. We then use a clustering procedure similar to the process described in 4.3.1. Note that although the model categorizes the extracted words as part of the output, the category names are arbitrary and inconsistent among examples. For example, the model can categorize a word *Italian* once as *food_type* and on another occasion as *nationality*. Therefore, we do not use them to obtain the slot groups directly. This leaves us with a set of slot candidates again used as input to our pipelined method.

4.4 Selection of slot candidates

In the previous step, we obtained a superset of all the slot candidates using weak supervision from the tagging models. Subsequently, we need to identify domain-relevant slots based on candidates provided by the automatic annotation. To achieve this, we design an iterative slot discovery procedure – in each iteration, we: (1) merge similar candidates, (2) rank candidates' relevance and eliminate irrelevant ones. Once no more frames are eliminated, the process stops and we obtain slot labels, which are used to train a slot tagger (see Section 4.5).

We refer to the automatically tagged tokens as *(slot) fillers*, and the tags are considered slot candidates. To be able to select relevant candidates, we need to represent them in continuous space. We use word embedding vectors and compute *slot embeddings* $e(s_k)$ for each distinct slot candidate s_k as word embedding averages over all respective slot fillers, weighted proportionally by filler frequency. The merging step requires the slot embeddings to be re-computed after each iteration. We will now describe the individual steps.

4.4.1 Candidate Merging

Since automatic annotation may have a very fine granularity, multiple slot candidates often capture entities/objects of the same type. This is the case for frame-semantic annotation, which we mostly use in our experiments. With a frame parser, for instance, the frames *Direction* and *Location* both relate to the concept of *area*, which can be represented as a single slot. Thus, we must merge similar slot candidates subsets $s_1 \dots s_n$ under a single candidate. We further use a syntactic parser to obtain dependency relations in which the slot fillers appear in the data and use this information to get more accurate similarity scores. We

[2]https://openai.com/blog/chatgpt

measure the similarity of slot candidates s_1, s_2 as:

$$\mathrm{sim}(s_1, s_2) = \mathrm{sim}_e(e(s_1), e(s_2)) + \mathrm{sim}_{\mathrm{ctx}}(s_1, s_2) \tag{4.1}$$

where sim_e is a cosine similarity and $\mathrm{sim}_{\mathrm{ctx}}(s_1, s_2)$ is a normalized number of occurrences of s_1 and s_2 with the same dependency relation. If the similarity exceeds a pre-set threshold T_{sim}, the candidates are merged into one.

4.4.2 Candidate Ranking and Selection

In this step, we aim to eliminate irrelevant slot candidates and exclude them from the selection process. To achieve this, we rank the slot candidates with respect to their importance computed from the data. We hypothesize that different slots are likely to occur in different contexts (e.g., addresses are mentioned more when the system provides information to the user rather than stated by the user). Some slots can occur rarely but still be relevant. However, such rare slots would be overshadowed by more frequent slot candidates. To preserve relevant slots that only occur in rarer contexts, we cluster the data into multiple clusters and then rank the candidates within each cluster separately. Finally, we filter the candidates according to a threshold. Specifically, we consider all candidates with a score higher than the chosen threshold relevant and select them for the next iterations. The threshold is determined as an α-fraction of a given cluster mean where α is chosen empirically. If a slot candidate is selected in at least one of the clusters, it is considered viable overall.

Clustering the data The data clustering step aims to distinguish contexts in which the candidates appear. We simplify the notion of context to the head verb connected with the respective slot filler word. We process the data with a generic semantic role labeling tagger to obtain verb dependency relations. Each occurrence of a filler is thus associated with a *head verb* whose semantic argument the corresponding word is, if such exists. To give an example, consider *verb-filler* pairs such as *want-chinese* or *reserve-hotel.* We then compute embeddings of the formed pairs. To do this, we take an embedding of both the head verb and the fillers and average them. The pairs are then clustered using agglomerative (bottom-up) hierarchical clustering with average linkage according to the cosine distance of their embeddings. Note that fillers for the same slot candidate may end up in multiple clusters. This does not mean that the respective slot candidate is split – it is just ranked for relevance multiple times (with respect to multiple contexts). The process stops when a predetermined number of clusters is reached.

Candidate Ranking criteria We need a function that computes a score for each candidate to rank the candidates. Since it is unclear how to compute the score, we combine multiple attributes to compute the final score. Specifically, we compute the following characteristics for each candidate:

- **Frequency** $\mathrm{frq}(s)$ is used since candidates that occur frequently in the data are likely important.
- **Coherence** $\mathrm{coh}(s)$ is the average pairwise similarity of all fillers' embeddings:

$$\mathrm{coh}(s) = \frac{\sum\limits_{(v,w)\in C_s^2} d_{\cos}(e(v), e(w))}{|C_s^2|} \tag{4.2}$$

  where C_s^2 is a set of all pairs of fillers for the slot candidate s. We follow Chen et al. (2014)'s assumption that fillers with high coherence, i.e., focused on one topic, are good slot candidates.
- **TextRank** (Mihalcea and Tarau, 2004) is a keyword extraction algorithm similar to the well-known PageRank (Page et al., 1999). It constructs a graph where nodes represent words and edges represent their co-occurrence. The dominant eigenvector of the adjacency matrix of this graph then gives the individual words' scores.

The final score is a simple sum of rankings with respect to all three scores. For TextRank and frequency, we use a placeholder representing the slot candidate instead of the respective fillers. Therefore, we obtain scores relevant to candidates rather than the individual words.

4.5 Training a standalone tagger

The steps described in Sections 4.2, 4.3 and 4.4 can give us a good set of dialogue slots. However, directly using the merged and filtered slots may result in low recall since the original annotation models used as weak supervision are not adapted to our specific domain. Therefore, we use the obtained labels to train a new, domain-specific slot tagger to improve performance. The tagger has no access to better labels than those derived by our method; however, it has a simpler task, as the set of target labels is now much smaller, and the domain is much narrower.

The motivation for training a tagger is two-fold. First, its usage makes it possible to discard a dependency on the candidate identification models in runtime. Thus, the method is simpler to apply. Second, although the selection process can yield a good set of slot candidates, we experimentally discovered that the quality of the taggers used for initial input labeling can be insufficient, especially for some domains. Therefore, directly using the merged and filtered slots may result in low recall since the original annotation models used as weak supervision are not adapted to our specific domain.

We model the slot tagging task as sequence tagging, using a convolutional neural network that takes word- and character-based embeddings of the tokens as the input and produces a sequence of respective tags (Lample et al., 2016).[3] The output layer of the tagger network gives softmax probability distributions over possible tags.

Increasing recall To further increase recall, we add an inference-time rule – if the most probable predicted tag is ‘O’ (i.e., no slot) and the second most probable tag has a probability higher than a preset threshold T_{tag}, the second tag is chosen as a prediction instead. As we discuss in Section 4.8, this threshold is crucial for achieving substantial recall improvement.

Tagging model robustness We only use 10% of the original in-domain training set (with labels from Section 4.2) to train the slot tagger model. The rest of the training set is used for a grid search to determine model hyperparameters (hidden layer size, dropout rate, and T_{tag} threshold). We choose the parameters that yield the best F1 score when compared against the automatic slot discovery results (i.e., no manual annotation is needed here; the aim is a good generalization and improved robustness of the resulting model).

4.6 Experimental setup

In this section, we provide a quantitative analysis of the results with respect to the NLU performance and quality of the discovered slots. We also evaluate the application of this method as a module in the end-to-end dialogue system model.

[3] `https://github.com/deepmipt/ner`

Datasets We use multiple datasets to evaluate the proposed method extensively and gain more insights. The datasets vary in several properties, like domain count or collection process. This means we can compare the results on different data distributions and tasks with different complexities. For more detailed dataset descriptions, please refer to Section 2.6. Here, we provide only a concise list with basic descriptions.

- **CamRest676** (**CR**) is dataset of 2,744 user utterances, all in the restaurant reservation domain.
- **Cambridge SLU** (**CS**) is in the same domain as CamRest676 but only focuses on NLU annotation of 10,569 user utterances.
- **MultiWOZ** is a multi-domain dialogue corpus. For experiments in this chapter, we picked only single-domain dialogues from two domains – hotel reservation and attraction recommendation – to form **WOZ-hotel** (**WH**) with 14,435 utterances, 9 slots, 3 intents and **WOZ-attr** (**WA**) with 7524 utterances, 8 slots and 3 intents respectively.
- **ATIS** (**AT**) is focused on flight search and contains nearly 5,000 utterances.

3rd party models As sources of weak supervision providing slot candidates, we mainly use the frame semantic parsers *SEMAFOR* (Das et al., 2010) and *open-sesame* (Swayamdipta et al., 2017) – a union of labels provided by both parsers is used in all our setups. In addition, to explore combined sources on the named-entity-heavy ATIS dataset, we include a generic convolutional NER model provided by SpaCy.[4] To provide features for slot candidate merging and selection, we use AllenNLP (Gardner et al., 2017) for SRL and FastText (Bojanowski et al., 2017) as pre-trained word embeddings.

4.6.1 Training details

- Slot merging and selection parameters were set heuristically in an initial trial run on the CamRest676 data and proved stable across domains.
- Slot tagger hyperparameters are chosen according to grid search on a portion of the training data.
- Since the models are rather small concerning the number of parameters, it is sufficient to use a regular desktop PC. In our experiments, we require about 4 GB of RAM and use Intel Xeon E5-2630 v4 CPUs.

[4]`https://spacy.io`

- Our slot candidate selection step takes roughly 1 hour. The tagger model is lightweight, with only 150k parameters. Its training requires 10-30 minutes, depending on the exact configuration and data size.
- We conduct a hyperparameter search using a basic grid search algorithm. We tested hidden size values $\in [50, 200]$, dropout $\in [0.5, 0.85]$ and the threshold $T_{\text{tag}} \in [0.05, 0.3]$. Therefore, we ran $4 \times 8 \times 6 = 192$ search trials.
- The best parameters were determined by tagger accuracy on the validation set: hidden_size = 250, dropout = 0.7, $T_{\text{tag}} = 0.3$, $T_{\text{sim}} = 0.9$.
- For data without explicit *train:validation:test* splits we use ratio of sizes *8:1:1*.

4.6.2 Evaluated systems

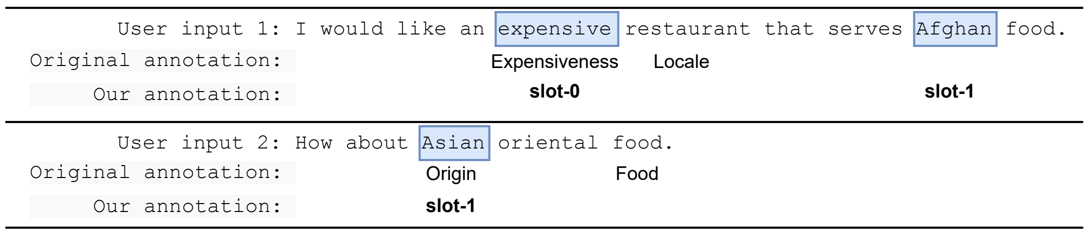
We test multiple variants of our system. This gives us an idea about the contributions of all the individual methods we propose. Here we give an overview of all the system variants:

- *Ours-full* is the full version of our method (full annotation setup and trained slot tagger).
- *Ours-nothr* does not use the recall-increasing second-candidate rule in the slot tagger.
- *Ours-notag* excludes the slot tagger. This means that the outputs of input taggers are used directly to annotate the data.
- *Ours-nocl* further excludes the clustering step; slot candidate ranking and selection is performed over all candidates together.

We also compare to previous work of Chen et al. (2014)[5]. This method is similar to the variant *Ours-nocl* but does not merge similar frames and uses different ranking criteria. Essentially, they use the outputs of the input tagger directly after the selection step without further processing.

To put our results into perspective, we also include two supervised models for comparison: *Tag-supervised* is the same model that we use as our slot tagger (see 4.5), but it is trained on supervised data with all ground truth labels available. The other supervised baseline is called *Dict-supervised.* It uses a simple dictionary of slot fillers obtained directly from the training data. We use straightforward word matching based on regular expressions to tag occurrences of these values.

[5]We use our re-implementation of their approach.


```
        User input 1: I would like an expensive restaurant that serves Afghan food.
Original annotation:                  Expensiveness   Locale
     Our annotation:                      slot-0                                slot-1

        User input 2: How about Asian oriental food.
Original annotation:             Origin        Food
     Our annotation:             slot-1
```

Figure 4.4: A sample of a dialogue from CamRest676 data, with labels from a frame-semantic parser (middle) and our slot tagger (bottom). Although "Afghan" food is not in the frame parser output, our tagger could recognize it. The second utterance successfully captures the change in value for slot-1 (corresponding to food type). This shows that our model can categorize entities (both "Afghan" and "Asian" relate to the same slot).

Apart from evaluating the tagging performance with respect to NLU, we are also interested in the intrinsic evaluation of the verb-slot pair clusters formed for slot ranking. Specifically, we ask how well these clusters are formed and if they are meaningful. We compare to gold-standard intent annotation with respect to the following baselines: (1) a majority baseline (assigning the most frequent intent class to all instances), and (2) a simple method that represents the utterances as averages of respective word embeddings and performs sentence-level intent clustering. All the slots in a given utterance are assumed to have the same intent.

4.7 Evaluation

We need a way of comparing the predicted structure to the ground truth to evaluate the discovered set of slots. Therefore, we construct a handcrafted mapping between our discovered slots and the respective ground-truth slots. Importantly, this mapping is only needed for evaluation; **our method does not depend on it**. The mapping is domain-specific, but it is very easy to construct even for an untrained person – the process takes less than 10 minutes for each of our domains. It amounts to matching slots from the domain ontology against slots output by our approach, represented by the FrameNet labels. We provide an example of such a reference mapping in Table 4.3.

We use some common metrics described in Section 2.7 for quantitative evaluation. Specifically we use **Intent Accuracy**, **Joint Goal Accuracy** and **Entity Match Rate.** We also use some metrics specific to this part of the work:

***Ours-full* output**		**CambridgeSLU ontology**
Expensiveness	$\mapsto$	Pricerange
Origin + People_by_origin	$\mapsto$	Food
Direction + Part_orientational	$\mapsto$	Area
Contacting + Artifact	$\mapsto$	Phone
Locale_by_use	$\mapsto$	Type

Table 4.3: An example of reference mapping between the output of *Ours-full* represented by FrameNet labels (left) and ground-truth CambridgeSLU ontology (right). Frames merged by our method are shown on a single line, separated by "+".

- **Slot F1 score**: To reflect slot tagging performance, we measure precision, recall, and F1 for every slot individually. An average is then computed from slot-level scores, weighted by the number of slot occurrences in the data. We measure slot F1 on standalone user utterances (slot tagging) and in the context of a dialogue system (dialogue tracking).
- **Slot-level Average Precision (AP)**. The slot candidates picking task is a ranking problem, and we use the *average precision* metric following Chen et al. (2014). Considering a ranked list of discovered slots $l = s_1, \ldots, s_k, \ldots, s_n$ we compute AP:

$$AP(l) = \frac{\sum_{k=1}^{n} P@k(l)\mathbb{1}_k}{\#\,\text{mapped slots}} \tag{4.3}$$

  where $\mathbb{1}_k$ is an indicator function that equals one if slot k has a reference mapping defined and $P@k(l)$ is precision at k of the ranked list l.
- **Slot Rand Index (RI)** is a clustering metric used to evaluate slot candidate merging. RI is the proportion of pairs of slot candidates correctly assigned to the same or different slots (following the reference mapping).[6]
- **Normalized Mutual Information (NMI)** is the mutual information between two clusterings normalized into the (0, 1) interval. Thanks to the normalization, it is suitable for comparing two clusterings with different numbers of clusters.

For slot tagging and ranking evaluation, we sampled a random data order 50 times and performed 5-fold cross-validation for each permutation. For the dialogue generation evaluation, we trained the models 100 times and used averaged results. All results are given with 95% confidence intervals.

[6]We compute RI on a union of labels with a ground-truth slot mapping and all labels selected by our method. Labels without ground-truth mapping are assumed to form single-item "pseudo-slots".

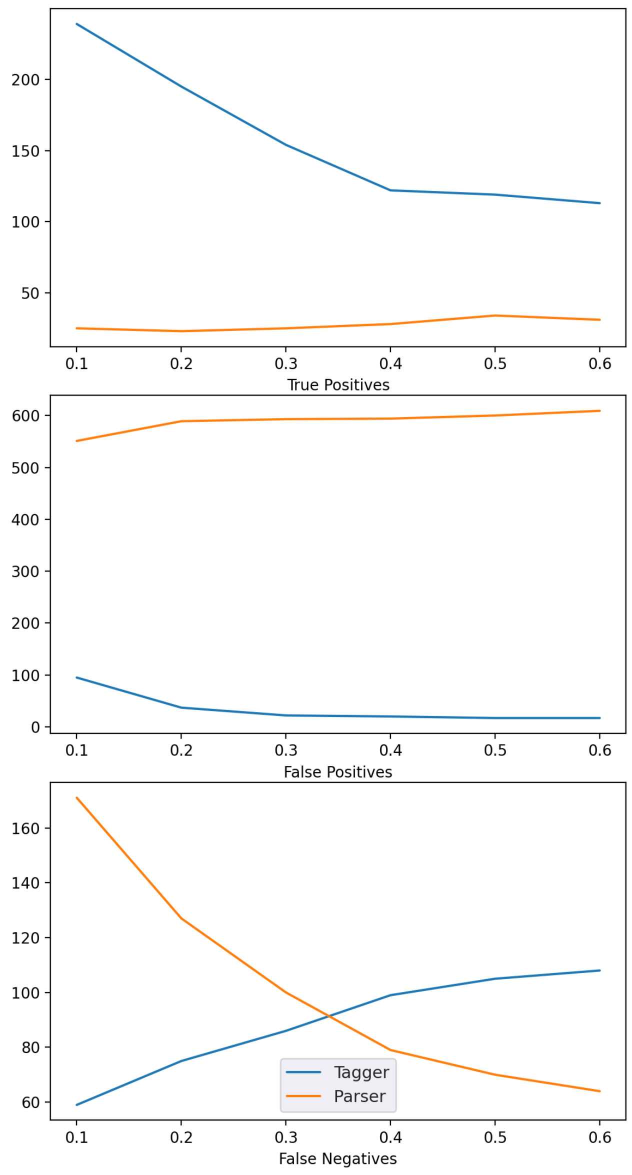


Figure 4.5: The comparison of outputs of our tagger and the parser. The plots show a number of cases in which the respective approach encounters more TPs, FPs, or FNs than the other.

4.8 Results

We evaluate our approach to slot discovery by comparing the resulting slot labels to gold-standard supervised slot annotation.

Slot tagging is evaluated in Table 4.4. *Ours-full* (slot selection + trained tagger) outperforms all other approaches by a large margin, especially regarding recall.

The performance cannot match the supervised models, but it is not far off in some domains.[7] Chen et al. (2014)'s method has a slightly higher precision, but our recall is much higher than theirs. Note that Chen et al. (2014) do not reduce the set of candidates. They only rank them so that a manual cut-off can be made. In contrast, our method reduces the set of candidates significantly. A comparison between *Ours-notag* and *Ours-full* shows that applying the slot tagger improves both precision and recall. Tagger without the threshold decision rule (*Ours-nothr*) mostly performs better than the parser; however, using the threshold is essential to improve recall. Experiments on ATIS with NER as an additional annotation source proved that our method can benefit from it. As discussed above, using the trained tagging model is crucial to improve the recall of our method. In Figure 4.5, we compare the results with and without the tagger. We change the value of the prediction threshold and measure the number of cases in which the tagging model encounters more true positives, false positives, or false negatives, respectively. As the results show, lowering the threshold increases the number of cases in which the tagger finds more correct slot values (and therefore improves recall), while it does not affect the number of false positives much (and therefore retains precision).

Performance of unsupervised candidate identification We separately evaluate the setup in which we used unsupervised sources of candidate identification described in Section 4.3. In Table 4.5, we present the performance of the overall pipeline for various candidate identification methods with and without applying our full pipeline (see 4.1). We can see that applying our pipeline improves the performance, especially the overall recall. We also compare the pipeline performance with various candidate identification methods in Figure 4.6. This comparison shows that the FrameNet-based candidate identification mechanism outperforms unsupervised variants.

[7]Note that our measurements of slot F1 only consider the 'O' tag as negative (the average is computed over slots only). This results in lower numbers than those reported in literature (Goo et al., 2018), but we believe this reflects the actual performance more accurately.

method ↓ / **dataset**→	**CS**	**WH**	**WA**	**AT**
Tag-supervised*	0.724 ± .003	**0.742** ± .008	**0.731** ± .002	**0.848** ± .003
Dict-supervised*	**0.753** ± .005	**0.750** ± .018	0.665 ± .003	0.678 ± .002
weak supervision →	frames	frames	frames	frames,NER
Chen et al.	0.590 ± .001	0.382 ± .001	0.375 ± .001	0.616 ± .001
Ours-nocl	0.393 ± .011	0.122 ± .001	0.266 ± .008	0.677 ± .002
Ours-notag	0.664 ± .007	0.388 ± .002	0.383 ± .002	0.648 ± .003
Ours-nothr	0.569 ± .031	0.485 ± .032	0.435 ± .002	0.698 ± .004
Ours-full	**0.692** ± .008	**0.548** ± .004	**0.439** ± .001	**0.710** ± .002

Table 4.4: F1 score values with 95% confidence intervals for slot tagging performance comparison among different methods. The measures are evaluated using a manual slot mapping to the datasets' annotation, which is unnecessary for the methods. *Note that supervised setups are not directly comparable to our approach.

method	**Precision**	**Recall**	**F1**
Mem2Seq	0.72	0.22	0.31
LLM-ChatGPT	0.76	0.23	0.35
LLM-Tk-instruct	0.78	0.08	0.14
Mem2Seq+Ours	0.92	0.26	0.38
LLM-ChatGPT+Ours	0.67	0.29	0.39
LLM-Tk-instruct+Ours	0.49	0.17	0.22

Table 4.5: Cluster assignment accuracy of our methods if we interpret the clustering as user intent detection. *Majority* is a majority baseline, and *Embedding* refers to an average sentence embedding clustering approach.

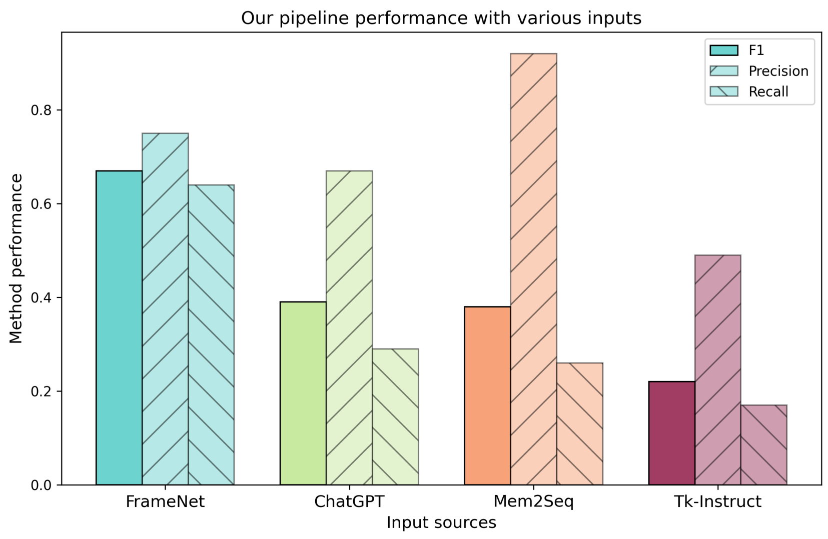


Figure 4.6: Comparison of our pipeline performance with different input sources. Note that FrameNet achieves both the best performance and the best trade-off between Precision and Recall

4.8.1 Error analysis

We conducted a manual error analysis of slot tagging to gain more insight into the output quality and sources of errors. We found that the tagger can generalize and capture unseen values.

One source of errors is the relatively low recall of the frame-semantic parsers. We successfully addressed this issue by introducing the slot tagger. However, many slot values remain untagged. This is expected as the input linguistic annotation quality inherently limits our method's performance. The candidate merging procedure causes another error (see also below). Due to frequent co-occurrence, two semantically unrelated candidates might be merged, and therefore, some tokens are wrongly included as respective slot fillers. Nevertheless, the merging step is required to obtain a reasonable number of slots for a dialogue domain.

Our approach does leave some room for improvements, especially regarding the consistency of results across different slots, which can be imbalanced.

dataset	**price**	**area**	**request**	**type**	**food**	**day**	**people**	**stars**	**stay**
CR	0.54	0.76	0.76	–	0.59	–	–	–	–
CS	0.63	0.84	0.48	0.81	0.64	–	–	–	–
WH	0.21	0.52	0.11	0.13	–	0.15	0.82	0.82	0.34

Table 4.6: Per-slot F1 scores of the *Ours-full* method evaluated on selected datasets with slot intersection. For some slots, the performance varies a lot among datasets due to different ranges of values and contexts. The measures are evaluated using a manually designed slot mapping to the datasets' annotation, which is unnecessary for the methods.

method	**CR**	**CS**	**WH**	**WA**	**AT**
Chen et al.	0.315 ±.002	0.272 ±.001	0.269 ±.001	0.393 ±.002	**0.267** ±.003
Ours-nocl	**0.519** ±.003	0.376 ±.003	0.069 ±.074	0.176 ±.016	0.069 ±.008
Ours-full	**0.520** ±.004	**0.400** ±.003	**0.317** ±.008	**0.403** ±.006	0.208 ±.018

Table 4.7: Slot candidate ranking average precision for all datasets

Slot candidate ranking results are given in Table 4.7. Our pipeline significantly outperforms Chen et al. (2014)'s approach on 4 out of 5 datasets. We can also see that the slot-verb pairs clustering step is important – in the ablation experiment where we do not perform clustering (*Ours-nocl*), performance falls dramatically on the WOZ-hotel, WOZ-attr, and ATIS data. This is because, without the clustering step, many context-irrelevant slot candidates are considered, hurting performance.

In addition, we include a detailed evaluation of the contribution of the individual slot candidate ranking scores. Results in Table 4.8 suggest that all of our proposed scores improve the performance.

Slot merging evaluation is shown in Table 4.9. Although candidates in the CamRest676 data are merged into slots reasonably well, other datasets show a relatively low performance. The low RI scores result from errors in candidate ranking, which wrongly assigned high ranks to some rare, irrelevant candidates. These candidates do not appear in the reference mapping and are assumed to form singular "pseudo-slots". However, they are typically joined with similar candidates in the merging process. This leads to many pairs of candidates merged into one slot by our approach but appearing separately in the reference mapping. Nevertheless, this behavior barely influences slot tagging performance as the candidates are rare.

configuration	F1 score
Ours-full	**0.663** $\pm$ 0.012
Ours -frq	0.600 $\pm$ 0.008
Ours -coh	0.582 $\pm$ 0.012
Ours -TextRank	0.514 $\pm$ 0.006

Table 4.8: Ablation study of slot ranking features on CamRest676. The full model is compared to variants left out of the scores.

	method	CR	CS	WH	WA	AT
RI	Rnd	0.466	0.268	0.155	0.153	0.178
	Ours	0.587	0.319	0.168	0.188	0.171
NMI	Rnd	0.212	0.137	0.061	0.128	0.171
	Ours	0.359	0.207	0.101	0.117	0.194

Table 4.9: Slot merging evaluation using RI and NMI on selected datasets, comparing our approach (*Ours*) with a random baseline (*Rnd*).

Clustering evaluation: Table 4.10 suggests that our clustering performs better than simple baselines and can yield useful results for intent detection. Nevertheless, intent detection is more complex and presumably requires more features and information about the dialogue context, which we reserve for future work. The complexity is also suggested by the naive embedding clustering performing worse than the majority baseline in 4 out of 5 cases.

method	**CR**	**CS**	**WH**	**WA**	**AT**
Majority	0.592	0.530	0.883	0.612	**0.727**
Embedding	0.535	0.551	0.873	0.595	0.705
Ours	**0.705**	**0.613**	**0.882**	**0.699**	0.677

Table 4.10: Cluster assignment accuracy of our methods if we interpret the clustering as user intent detection. *Majority* is a majority baseline, and *Embedding* refers to an average sentence embedding clustering approach.

4.9 Dialogue generation application

To verify the usefulness of the labels discovered by our method, we use them to train and evaluate an end-to-end task-oriented dialogue system. We choose Sequicity (Lei et al., 2018b) based architecture for our experiments, an LSTM-based encoder-decoder model that uses a system of copy nets and two-stage decoding. First, it decodes the dialogue state so the database can be queried externally. Sequicity generates the system response conditioned on the belief state and database results in the subsequent step. This architecture works with a flat representation of the dialogue state, i.e. the state is represented as a sequence of tokens – slot values.

The basic architecture is further extended by Jin et al. (2018). They propose to model the dialogue state explicitly in a semi-supervised way. They extend the end-to-end encoder-decoder dialogue response generation model of Lei et al. (2018b) by introducing an additional decoder with access to posterior information about the system response. This allows them to train a state representation with a reconstruction loss on unsupervised examples, using the state as a limited memory for essential concepts (roughly corresponding to slots). Their method can be applied fully unsupervised, but it still requires some in-domain annotations to perform well. The default Sequicity model uses gold-standard dialogue state annotation. However, a compatible state representation is directly obtainable from our labels simply by concatenating the labels aggregated in each turn from user utterances. Whenever a new value for a slot is found in user input by our tagger, it is either appended to the state representation or replaces a previous value of the same slot.

We run three versions of the Jin et al. (2018)'s model: *Jin et al. supervised* is a model that is trained fully on supervised data to provide perspective on the achieved performance. *Jin et al. unsupervised* is, on the contrary, fully unsupervised, i.e. we provide no labeled examples during the training phase to give a fair comparison against our model. Finally, *Jin et al. weak-labels* does not use supervised labels but presents labels obtained by our method.

method	Slot F1	Joint Goal Accuracy	Entity Match Rate
Jin et al. supervised	0.967 ± .001	0.897 ± .002	0.869 ± .004
Jin et al. unsupervised	0.719 ± .002	0.385 ± .003	0.019 ± .002
Jin et al. weak-labels	0.709 ± .011	0.335 ± .008	0.269 ± .012
Ours-full (unsupervised)	**0.756** ± .004	**0.465** ± .007	**0.368** ± .008

Table 4.11: Evaluation on the downstream task of dialogue generation on CamRest676 data. We evaluate with respect to three state tracking metrics. The best results in an unsupervised setting are presented in bold.

4.9.1 Results

We explore the influence that our labels have on sequence-to-sequence dialogue response generation in an experiment on the CamRest676 data (see Table 4.11). We can see that our method provides helpful slot labels that improve dialogue state tracking performance. Our approach significantly improves all metrics compared to Jin et al. (2018)'s system used in a fully unsupervised setting. We achieve better results than Jin et al. (2018)'s system, especially regarding entity match rate, suggesting that our model can provide consistent labels throughout the dialogue. To make a fair comparison, we further evaluate Jin et al. (2018)'s system in a setting where it can learn from the labels provided directly by weak supervision (i.e., the frame-semantic parser, not filtered by our pipeline). We observe an improvement in entity match rate, but it does not match the improvement achieved with our filtered labels. Surprisingly, slot F1 and joint goal accuracy even decreased slightly, which suggests that label quality is important and the noisy labels obtained directly from weak supervision are not useful enough.

4.10 Conclusion

In this chapter, we propose a pipeline method that analyzes a set of clusters of semantically related values and yields a set of candidates for dialogue slots relevant to the given task. The methods can take unstructured conversation data as inputs and require no data labels as inputs.

We show that our pipeline can work with various candidate identification methods and improve their performance regardless of the particular setup. The method also compares favorably to similar proposed approaches.

The analysis shows that our method is rather sensitive to the quality of input clusters, i.e. the performance of the initial source of the candidate annotation (weak supervision, Mem2Seq, LLMs). Our method is easy to apply and provides a reasonable performance, improving with the increasing quality of input slot candidates.

For future work in this area, we can consider applying better methods for text representations and parsing, which would likely improve the qualitative performance. It is also desirable to address the need for setting empirical thresholds, especially in the slot candidate selection and merging phase, exploiting the power of pre-trained language models. Overall, the method is model-agnostic in the sense that various components of the pipeline can be swapped with better-performing models, which would likely improve the quantitative performance.

5 Dialogue modeling with less supervision

Dialogue modeling is a complicated task requiring the ability to communicate in a natural language and handle discrete decision processes representing the conversation logic. Various architectures have been proposed over the years (see Sections 2.5, 3.2), mostly relying on explicit data annotation on multiple levels to guide the model training process. One of the biggest challenges is to model task-oriented dialogue that requires interaction with external interfaces such as databases or API services. This aspect puts a hard constraint on the dialogue system architecture – it requires some explicit representation that allows one to communicate with external systems. Achieving this in a fully unsupervised setting is challenging due to lack of model guidance in the form of structured labels. However, we explore multiple viable approaches to this problem in this chapter.

First, we discuss the challenges of unsupervised task-oriented dialogue modeling in more detail in Section 5.1. Next, we propose our architecture that uses latent representations and explore its abilities and performance in Section 5.2.1. Finally, we explore extensions to the latent variable models by proposing hierarchical architecture.

In this chapter, we present a modeling approach that we previously published in Hudeček and Dušek (2022) and propose alternative unpublished extensions of a different base architecture using latent representations (Lubis et al., 2022). Our experimental code is available on GitHub[1].

[1] `https://github.com/vojtsek/to-vrnn`

5.1 Challenges of TOD interaction with external interfaces under less supervision

There are multiple challenges to overcome when we try to model TOD without guidance in the form of labels (Section 2.5.2). This chapter addresses the challenges of action selection and interface interaction. To enable the model to interact with external sources of information via API, we annotated points in the training dialogues where interaction with an external API is needed. Therefore, the model can learn to construct the database queries ad-hoc using special outputs. A turn-level annotation of database queries would represent a similar amount of annotation used in supervised training. It thus would not lead to our desired setting, where the model is trained without labeled data. To support database access while avoiding costly turn-level annotation, we follow Bordes et al. (2017) and insert sparse database queries and results into the training data, forming special dialogue turns. Specifically, we identify turns that require database results, e.g. to inform about entity attributes or the count of matching entities, and insert a query-result pair in front of those turns (see Table 5.1). We argue that this is the minimal level of supervision required to successfully operate a task-oriented system with database access; it is significantly lower than the full dialogue-state supervision used by most systems. Moreover, this kind of supervision is readily available in a realistic setting (such as using call center recordings and agent activity logs). In practice, database queries are only inserted for 24% turns on average. Per-dataset query counts are 36%, 23%, and 11% for CamRest676, MultiWOZ, and SMD, respectively. Note that this approach still covers the task of an explicit state tracker since the necessary entity values are provided when needed. Database query results can be stored and used in follow-up questions to maintain consistency. .

Turn 1	**user:**	`Is there a` **`moderately priced`** `restaurant serving` **`italian`** `food anywhere in town?`
	system gold:	`query (italian moderate)`
	action:	`QUERY()`
Turn 2	**user/database:**	`pizza express, Regent Street City Centre, 01223 324033, C.B 2, 1 D.B, centre`
	system gold:	`Pizza express serves Italian food and is located in the town centre and is in the moderate price range .`
	action:	`OFFER()`
Turn 3	**user:**	`what is the` **`address`** `and` **`phone number`** `?`
	system gold:	`their address is Regent Street City Centre. Their phone number is 01223 324033. can I help with anything else ?`
	action:	`GIVE_DETAILS()`

Table 5.1: An example dialogue drawn from the CamRest676 validation set illustrates our injection of database interaction turns. We show the user input, inserted database queries/results (color-coded), the gold-standard system response, and system action annotation based on manual rules. A database query is constructed in the first turn, and the second turn illustrates how the result is retrieved and fed as input.

5.2 Task-Oriented dialogue with TO-VRNN

The VRNN model architecture (see Section 2.3.1) is designed to model sequences of observations coupled with latent states. It is a generative model that can learn the conditional generative distribution of observations given the state. Moreover, although VRNN does not require a fixed set of states, it can be adjusted to model discrete states. The VRNN is great for modeling the discrete decision processes behind conversation exchanges, thanks to the above-mentioned properties. However, we propose some extensions for task-oriented dialogue modeling to distinguish between the user and system roles and incorporate the possibility of handling interaction with external interfaces.

5.2.1 TO-VRNN model description

We extend the VRNN model introduced in Chapter 2.3.1 to fit the task-oriented setup. Figure 5.1 depicts our model's architecture. Following the original VRNN architectures (Section 2.3.1), we employ a turn-level RNN that summarizes the context in its hidden state. In each dialogue turn, we model user and system utterances with separate autoencoders to account for different user and system behaviors. This contrasts with the original model described in 2.3.1, which contains only one autoencoder in every time step. We can divide the processing of one turn into two stages. First, the user utterance, modeled with a standard vanilla autoencoder, is processed, and the last encoder hidden state $\varphi^u_{enc}(\mathbf{x}^t_u)$ provides the encoded representation used as an input for the next stage.

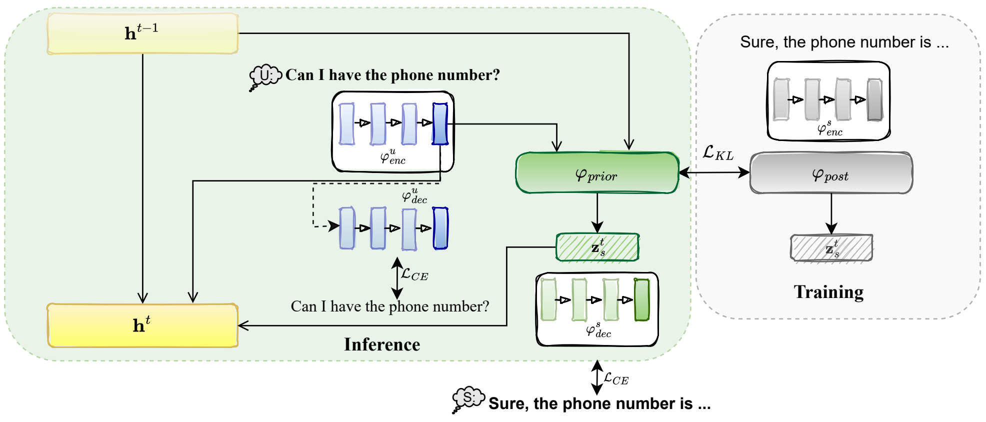


Figure 5.1: Visualization of our model architecture (one dialogue turn), described in Section 5.2.1. Yellow boxes represent the turn-level VRNN's hidden state h^t. The user utterance is represented as the last hidden state of the encoder network φ^u_{enc}, which is trained as an autoencoder along with the decoder φ^u_{dec}. The system utterance, encoded by the network φ^s_{enc}, is an input to the posterior network φ_{post} that helps to train the prior network φ_{prior} to construct meaningful latent variables $\mathbf{z}_s$, which initialize the system utterance decoder φ^s_{dec}. The training uses the whole architecture, including the posterior network φ_{post}, while only the part shaded in green is used for inference. $\mathcal{L}_{CE}$ stands for cross-entropy loss, $\mathcal{L}_{KL}$ for KL-divergence loss.

Next, the system part is used, which is realized by VAE with discrete latent variables $\mathbf{z}_s$ conditioned on the context RNN's hidden state $\mathbf{h}^{t-1}$ and the user utterance encoding $\varphi^u_{enc}(\mathbf{x}^t_u)$. The system module is trained as the VAE component in the VRNN model (2.3.1) and creates latent representations interpreted as system actions. This predicted latent variable is then used as an input to the system decoder that generates the realization of the latent action in the form of natural language. Our model can thus be seen as a VRNN extended by an additional encoder-decoder module for input pre-processing.

To finalize the turn-processing step, we need to save the information into the turn-level network so it becomes part of the encoded context. The turn-level network state update looks as follows:

$$\mathbf{h}^{t+1} = \text{RNN}([\varphi^u_{enc}(\mathbf{x}^t_u), \varphi_z(\mathbf{z}^t_s)], \mathbf{h}^t) \tag{5.1}$$

In other words, we use representations from both user and system modules to pass the information necessary to update the overall state.

For word-level encoding and decoding modules ($\varphi^u_{enc}, \varphi^s_{enc}, \varphi^u_{dec}, \varphi^s_{dec}$), we use an RNN with LSTM cells. We further experiment with attention (Bahdanau et al., 2014) over user encoder hidden states in the system decoder. We train the model by minimizing a sum of the cross-entropy reconstruction loss on user utterances and the variational lower bound (Equation 5.2) on system responses.

$$\begin{aligned} \log\ p(\mathbf{x}) \geq -\mathrm{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \\ +\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[\log\ p(\mathbf{x}|\mathbf{z})] \end{aligned} \tag{5.2}$$

The final objective is presented in Equation 5.3. There, ϕ and ψ represent the parameters corresponding to system and user modules, respectively. $\mathbf{x}_s$ and $\mathbf{x}_u$ are system and user utterances and $\mathbf{u}$ is the encoded representation of the user utterance.

$$\begin{aligned} \mathbb{L}_{TO-VRNN}(\phi,\psi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}_s)}[\log\ p_\phi(\mathbf{x}_s|\mathbf{z})] \\ -\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x}_s)||p(\mathbf{z})) \\ +log\ p_\psi(\mathbf{x}_u|\mathbf{u}) \end{aligned} \tag{5.3}$$

This objective directly extends the original training objective presented in 2.3.1.

When running in inference mode, only the prior distribution $p(\mathbf{z})$ is considered, which does not require the system utterance on the input. Therefore, the model can generate the system response only with a user utterance on the input.

Latent Variables We use a set of n K-way ($K = 20; n = 1, 3, 5$) categorical variables to achieve good interpretability, following Zhao et al. (2018). Each variable is represented as a one-hot vector of length K, and we use n such vectors. Multiple ways exist to incorporate trainable discrete variables into a differentiable network (see Chapter 2.3). We use the Gumbel-Softmax distribution and the reparameterization trick (Jang et al., 2017). We always take the item with the maximum probability concerning the predicted softmax distribution instead of sampling from it during inference.

5.2.2 TO-VRNN Experiments

In this section, we evaluate the quality of responses generated by our model. We also inspect the model performance concerning dialogue success. We focus on theoretical modeling and the feasibility of the proposed approach at this stage, which we believe is sufficiently demonstrated by corpus-based evaluation complemented by manual checks. Detailed interpretation of the learned representations follows in Section 5.2.4.

Data

We evaluate the model performance on three datasets: CamRest676 (Wen et al., 2017d), MultiWOZ 2.1 (MW)(Budzianowski et al., 2018b; Eric et al., 2020) and Stanford Multidomain Dialogues (SMD) (Eric et al., 2017) We use standard splits for MultiWOZ 2.1 and SMD. We split CamRest676 in the 8:1:1 ratio, following previous work. Detailed descriptions are given in Section 2.6.

Database queries To include database information in the dialogues as described in Section 5.1, we first identify all turns in the original datasets where database information is required, using handcrafted rules. These rules are very simple, and their creation requires minimal effort: whenever database results are provided in the data (based on simple pattern matches over system actions), we prepend a database query based on the ground truth state. The assumption is that these queries would naturally be available in a real-world scenario – database queries induced by human operators can be logged along with client-operator conversations. We then create special database query turns based on the respective state annotation (see example in Table 5.1). Note that database query parameters are the only annotation used to train our models apart from utterance texts; no other dialogue state annotation from the original datasets is used.

Experimental Setup

We evaluate two versions of our model: one that uses the attention mechanism (*attn*) and one without it (*noattn*). The number and size of the variables are set based on a few cursory checks on the training data. Our models use ten latent variables by default; we discuss the influence of the number of latent variables in Table 5.5. Since our approach is the first to be evaluated in a task-oriented setting with minimal supervision, comparing it to prior works is difficult. Setups with full dialogue-state supervision are not comparable, and dialog-state metrics are not applicable without turn-level supervision. Therefore, we compare our models to standard architectures, such as vanilla LSTM or Transformer encoder-decoder, predicting sequentially using the same amount of supervision as our approach. We also compare to the HRED/VHRED models, perhaps the closest prior work to our approach. To put the results into perspective, we also include scores for fully supervised state of the art on our datasets. However, note that these scores are not directly comparable.

Training details The model is trained with gradient descent using the ADAM optimizer. We set the hyperparameters according to the BLEU and perplexity results obtained during a grid search considering the system responses on the development set. The utterance encoder and decoder hidden sizes are 250; the context-LSTM hidden size is 100. The latent variables are 20-dimensional vectors; their number differs across experiments. For the RNN components, we use a dropout probability of 0.3. The total model size is 7,047,529 parameters. The training time is 3-8 hours using one GPU, depending on the dataset. The training is sensitive to some parameters, such as the Gumbel-softmax temperature, but otherwise, the model trains easily using conventional optimization methods. To address the issue of posterior collapse (see Section 2.3.3), we experiment with various approaches to KL annealing. We use a liner annealing schedule in our experiments, increasing the β annealing coefficient from $0 \rightarrow 1$.

Metrics We use common metrics described in Table 5.2. To evaluate the quality of individual responses, we compute BLEU score and perplexity on the test set. Without dialogue-state supervision, we cannot measure task-oriented metrics such *joint goal accuracy.* Therefore, we decided to measure dialogue success and entity match rate, which we adjusted to the minimally supervised case (details follow). We also measure the database query accuracy.

5.2.3 Results

Response quality

Our architecture performs substantially better than (V)HRED, which commonly fails to acquire the necessary knowledge, especially on larger datasets. The attention-based versions perform better on BLEU but lose slightly on perplexity. Comparing HRED and VHRED shows that using the variational approach improves overall performance. While the GPT-2 PLM outperforms our approach on perplexity, it is worse on BLEU score despite its huge capacity.

We compare to other relevant related works:

1. Shi et al. (2019) do not use their model for response generation, but they report a negative log-likelihood of approximately $5.5 \cdot 10^4$ when reconstructing the CamRest676 test set. Our *TO-VRNN-noattn* model obtained $0.87 \cdot 10^4$, which suggests a better fit of the data.[2]

[2]This comparison is only approximate since the exact data split is not described by Shi et al. (2019) – we can only use a test set of the same size, not the exact same instances.

2. Wen et al. (2017a) measure response generation BLEU score on fully delexicalized CamRest676 data. Their best-reported result is 24.60, while our model gets 27.23 (30.10 with attention).

Our models can generate relevant responses based on manual checks in most cases. As expected, only the models, including database turns, can predict correct entities (cf. Section 5.2.3). A relatively common error is informing about wrong slots, e.g., the model provides a phone number instead of an address or, even more frequently, provides wrong slot values (cf. Table 5.3).

Dialogue success The conventional definition of dialogue success or *success rate* reflects the ratio of dialogues in which the system captures all the mentioned slots correctly and provides all the requested information. We follow previous works (Nekvinda and Dušek, 2021b) and report corpus-based success scores instead of using a user simulator. However, measuring success rate without turn-level labels is not straightforward. We approximate tracking slot values turn-by-turn by checking for correct slot values upon database queries only, and we use this information to measure dialogue success. Note that this is not equivalent to having state tracking labels available at all turns, but we consider it a reasonable approximation given our limited supervision – database queries are crucial for presenting the correct entities to the user, which in turn decides the dialogue success.

The generated query attributes directly show the captured slots.

Success rate results are shown in Table 5.4. Our system is not competitive with a fully supervised model but outperforms the baselines (VHRED, GPT). Upon inspection, we see that the system can often recognize correct slots. However, it has difficulties capturing the right values. However, the scores are promising, considering the minimal supervision of our training.

Matching database entities Table 5.2 shows the performance of our models for entity matching. We observe that the model performance without the database information is poor. However, including the database information improves the performance substantially, especially in the case of CamRest676 data. The MultiWOZ data is much more complex – it contains more slots and multiple domains that can also be combined in an individual dialogue. Nevertheless, we can still observe an improvement when we include the database queries. We also note that using attention improves EMR substantially – the latent variables alone cannot hold all information about particular values (cf. Section 5.2.4).

model		CamRest676				MultiWOZ 2.1			
	db	BLEU	Ppl	MI	EMR	BLEU	Ppl	MI	EMR
LSTM	✗	3.90	5.34	–	–	0.92	8.23	–	–
Transformer	✗	4.98	7.72	–	–	0.95	6.95	–	–
GPT-2	✗	15.40	1.18	–	–	9.40	2.77	–	–
GPT-2	✓	13.89	1.80	–	–	9.56	2.43	–	–
HRED	✗	2.70	13.92	–	0.02	2.98	29.61	–	0.01
VHRED	✗	4.34	11.76	0.21	0.02	4.65	32.74	0.15	0.01
VHRED	✓	8.50	10.23	0.17	0.36	3.82	16.61	0.07	0.04
TO-VRNN-noattn	✗	12.98	4.64	0.29	0.01	7.18	9.16	**0.42**	0.02
TO-VRNN-noattn	✓	15.10	4.45	**0.34**	0.24	11.3	5.17	0.27	0.05
TO-VRNN-attn	✗	**17.37**	5.07	0.16	0.09	**12.28**	10.19	0.06	0.04
TO-VRNN-attn	✓	17.10	**4.23**	0.22	**0.81**	11.86	**6.03**	0.05	**0.08**
supervised *	✓	25.50	–	–	–	19.40	2.50	–	–

model		SMD		
	db	BLEU	Ppl	MI
LSTM	✗	1.62	7.84	–
Transformer	✗	1.53	6.33	–
GPT-2	✗	9.26	2.46	–
GPT-2	✓	4.54	2.02	–
HRED	✗	1.25	12.50	–
VHRED	✗	3.75	11.94	0.20
VHRED	✓	3.94	11.86	0.19
TO-VRNN-noattn	✗	7.35	6.18	**0.53**
TO-VRNN-noattn	✓	9.24	**6.01**	0.47
TO-VRNN-attn	✗	12.30	6.36	0.04
TO-VRNN-attn	✓	**12.40**	6.11	0.11
*supervised**	✓	14.40	–	–

Table 5.2: Model performance in terms of BLEU for generated responses, Perplexity (Ppl), and Mutual Information (MI) between the generated response and the latent variables $\mathbf{z}_s$. We do not evaluate the database-enriched models on SMD as SMD's database structure does not map easily to our annotation style. We measure MI only for the models that use latent variables explicitly. The *db* column indicates systems that use database information. *Note that the supervised state-of-the-art scores are not directly comparable, as the systems use full turn-level supervision.(Qin et al., 2020);

Turn 1	**user:**	`Is there a` **`moderately priced`** `restaurant serving` **`italian`** `food anywhere in town?`
	system gold:	`query italian moderate`
	system hyp:	`query` **`italian moderate`**
Turn 2	**user/database:**	`pizza express, Regent Street City Centre, 01223 324033, C.B 2, 1 D.B, centre`
	system gold:	`Pizza express serves Italian food and is located in the town centre and is in the moderate price range .`
	system hyp:	**`Pizza hut Cherry Hinton`** `is a Italian restaurant in the` **`centre`** `part of town`
Turn 3	**user:**	`what is the` **`address`** `and` **`phone number`** `?`
	system gold:	`their address is Regent Street City Centre. Their phone number is 01223 324033. can I help with anything else ?`
	system hyp:	`the phone number is` **`01223 324033`**`. Is there anything else I can help you with ?`

Table 5.3: Predicted outputs or our VRNN model for a conversation snippet presented in Section 5.1.

model	success	query acc.
CamRest676		
VHRED	0.21	0.91
TO-VRNN-noattn	0.28	0.84
supervised SotA (Peng et al., 2021b)	0.73	N/A
MultiWOZ		
TO-VRNN-noattn	0.10	0.98
supervised SotA (Peng et al., 2021b)	0.85	N/A

Table 5.4: Dialogue success and query accuracy comparison for VHRED, *TO-VRNN-noattn* using the database and a state-of-the-art supervised system.

	BLEU	Ppl	MI
TO-VRNN-noattn-1z	25.20	4.25	0.46
TO-VRNN-noattn-3z	26.80	4.24	0.26
TO-VRNN-noattn-5z	27.23	4.20	0.38
TO-VRNN-noattn-12z	29.83	4.12	0.35

Table 5.5: Evaluation of the model performance with different numbers of latent variables concerning automatic measures of BLEU, Perplexity (Ppl), and Mutual Information (MI) on the CamRest676 data.

Database query accuracy Further, we evaluate the accuracy of the database querying. This metric measures if the system queries the database at appropriate turns. The query's content is not considered in this case, as it is already considered in the success rate. On MultiWOZ, we get a near-perfect accuracy, while our approach loses to VHRED on CamRest676 (see Table 5.4). We hypothesize that different dialogue structures can cause such discrepancies among these two datasets. The dialogues in CamRest676 usually contain just zero or one query during a dialogue, so our model might generate more queries than necessary.

5.2.4 Latent Variable Interpretation

Explaining and interpreting the model behavior is crucial, especially in a setting without full supervision. Therefore, we design experiments to evaluate the model behavior and investigate whether the model captures salient dialogue features in the latent variables obtained during training on CamRest676 and MultiWOZ. While the latent variables seem to be mainly useful for interpretability or structure induction, they are likely also contributing to the performance as smaller latent spaces yield lower performance, as shown in Table 5.5.

Mutual Information

Finally, we compute mutual information (MI) between the generated text and latent variables as well as among the latent variables themselves (see Table 5.2).[3] We see that using attention dramatically affects the amount of MI between the latent variables and the generated text. Since attention bypasses the latent vectors, the decoder does not need to use them to store information.

[3]Since we measure MI between categorical variables, we quantize the continuous variables used in the VHRED model.

config	**CamRest676**	**MultiWOZ 2.1.**	
	gold	domain	action
random	0.17	0.14	0.09
majority	0.42	0.33	0.32
HRED	0.65	0.45	0.44
VHRED	0.52	0.36	0.32
GPT-2	0.65	0.60	0.55
TO-VRNN-attn	0.63	0.68	0.66
TO-VRNN-noattn	**0.75**	**0.70**	**0.69**
TO-VRNN-manual	0.59	–	–

Table 5.6: Accuracy of the domain and action decision-tree classifiers based on latent variables. For details about the manual annotation process, see Section 5.2.4.

Clustering the actions

First, we want to assess whether similar variables represent similar actions. We follow Zhao et al. (2018) and define utterance clusters according to the model's latent variables. As a baseline, we consider random cluster assignment. A stronger approach computes sentence representations using a BERT model tuned for sentence representations (Reimers and Gurevych, 2019a) and then clusters the obtained sentence embeddings using K-means clustering.

We then use the homogeneity metric (Rosenberg and Hirschberg, 2007) to evaluate the clustering quality for the reference classes determined by manually annotated system actions (which are used for evaluation only). Homogeneity reflects the information the clustering provides (and, by extension, the latent vectors used) and is normalized to the interval [0, 1]. The reason for choosing this metric is that it is independent of the number of labels and their permutations. If all clusters only contain instances of a single class, we get the maximum homogeneity.

We provide the results in Table 5.7. The clusters formed on the CamRest676 data are more homogeneous than MultiWOZ, likely because of the greater dataset complexity in the latter case. In all cases, our clusters are much more homogeneous than clustering formed by random assignment. We also compare favorably to a stronger baseline based on the clustering of the sentence representations. These results suggest that the representations formed by our model correspond much better to the true actions seen in the data.

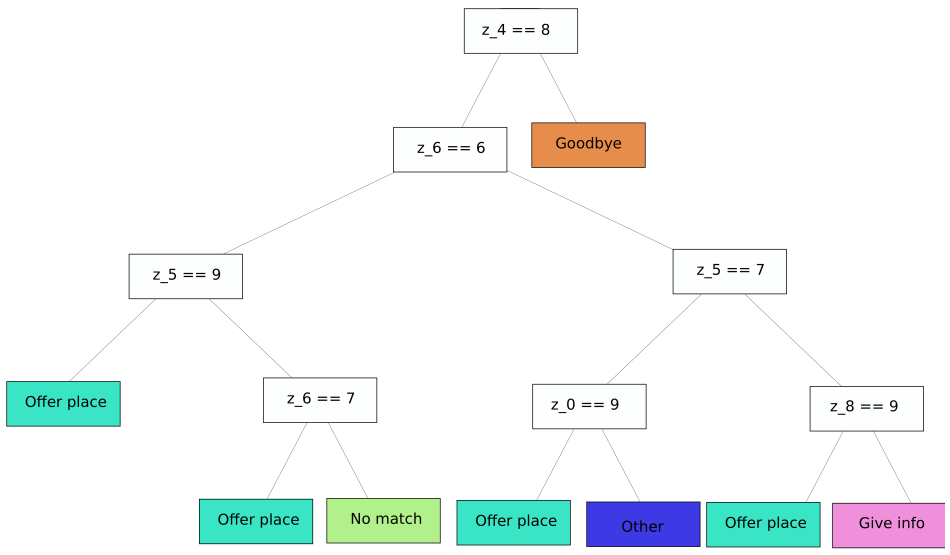


Figure 5.2: A visualization of a decision tree trained on the CamRest676 data to predict a system action from the contents of the latent variables. Each node represents a decision based on one latent variable value, and the leaf node colors represent different system actions. When the condition in a given node is fulfilled, the algorithm proceeds into the right subtree, left otherwise. For clarity, we limit the maximum tree depth to 4. The limit slightly lowers the accuracy – the pictured tree achieves an accuracy of 73% on the CamRest676 data.

Predictive power of the variables

To evaluate the predictive power of the obtained latent representations, we train a simple classifier that predicts the system action and current domain, using solely the obtained latent representations as input features. CamRest676 data does not include system action annotation. Hence, we manually designed a set of rules to determine system actions. An example of this rule-based action annotation is shown in Table 5.1. For MultiWOZ, we predict both system action and the domain of the utterance.

To put our results into perspective, we include several baselines: trivial random and majority class baselines and classifiers using representations obtained with other methods (HRED, VHRED, GPT). We use a decision tree (DT) classifier trained with the CART algorithm[4] and the *gini* split criterion due to its good interpretability. The results are shown in Table 5.6. Our classifier beats the random and majority baselines in all cases. It also outperforms classification based on (V)HRED and GPT representations. This demonstrates that our approach produces high-quality interpretable representations. We also observe that using attention harms the performance of the action classifier as it makes

[4]`https://scikit-learn.org/stable/modules/tree.html`

it possible for the models to bypass the latent variables. On CamRest676, the latent variables explain most of the annotated actions. Overall, we can observe that any hidden state taken from some trained model can explain some portion of the data. However, using our approach seems to perform better in this aspect. We also notice the influence of the number of latent variables on the performance. In general, increasing the number of latent variables leads to a substantial performance improvement, which suggests that all the variables contribute with relevant information (see Table 5.6).

The information about domains and system actions is stored in categorical variables. It can be extracted by a simple classification model such as the decision tree, which allows us to interpret and explain the behavior of our model. For illustration, in Figure 5.2, we plot a DT with limited depth that achieves 73% accuracy when predicting the system action on the CamRest676 data. The aim is that latent variables hold high-level information, such as intents, actions, or domains. This helps interpretability but is insufficient for generating appropriate and factually correct responses – here, we need to incorporate correct slot values. This detailed information is captured and carried over via the attention mechanism in *TO-VRNN-attn*. Potential alternatives are copy mechanisms (Lei et al., 2018a) or delexicalization on the generated outputs (Henderson et al., 2014; Peng et al., 2021b).

Manual interpretation

To explore the interpretability of our representations even further, we manually annotate the latent variables to obtain a simple handcrafted classifier. Specifically, we draw a set of pairs of utterances and corresponding latent representations from the validation set. We present the discrete representation vectors to an expert annotator to assign an action that each vector represents based on the sampled utterances. This way, we obtain a mapping from the space of latent vectors to actions. We then apply this mapping to predict actions on the test set (the *TO-VRNN-manual* entry in Table 5.6). Note that in this approach, we only allow assigning an action to a whole vector, unlike in the case of a decision tree classifier, which can take individual components into account. As the results show, this approach works well despite the above limitation.

Source	**CamRest action**	**MW action**	**MW domain**
TO-VRNN-noattn	0.65	0.34	0.39
sent-repr	0.45	0.33	0.30
random	0.20	0.02	0.01

Table 5.7: Homogeneity for *TO-VRNN-noattn* configuration using the database vs. a clustering of sentence representations and random baseline.

5.3 A Hierarchical Variational Model for TOD (HVTOD)

A structured representation of utterances in the dialogue with dialogue acts (Chapter 2.1) is hierarchical because it is a composition of several concepts that can constrain each other. Specifically, each utterance is characterized on the top level by its domain. Each domain is connected with some intents and system actions, typically further connected with certain slots. To reflect this hierarchical nature of the dialogue acts, we thus need some model that would structure its representations accordingly. Since we are interested in latent representations, VAE architecture is a good option thanks to reasons discussed in 5.1.

VAEs for hierarchical architecture were successfully applied in the computer vision domain (Vahdat and Kautz, 2020; Li et al., 2020b). We take inspiration from these works and apply the hierarchical variational autoencoder architecture to the dialogue domain. We hypothesize that this structure will allow the model to work with different levels of abstraction and thus represent the decision process more accurately and with greater detail. Note that there is no guarantee that the learned representations in different layers will also differ. We want to explore this question to see if the model can learn to leverage this structure by itself.

5.3.1 HVTOD model architecture

The proposed HVTOD architecture is based on the LAVA framework (Lubis et al., 2022, see Section 2.3.2). The authors propose an architecture in LAVA that uses latent representations of dialogue actions computed with VAE. They use context encoded with an RNN encoder as an input into the VAE. We extend the architecture so that instead of just one VAE. We use a set of VAEs stacked on top of each other to model the latent variables. The overall model architecture is depicted in Figure 5.1. In the input, we have a set of dialogues $\mathcal{D}$ where each dialogue d with n turns consists is represented as a sequence of altering system and user utterances $d = (s_1, u_1, ..., s_n, u_n)$. Similarly to LAVA, our model

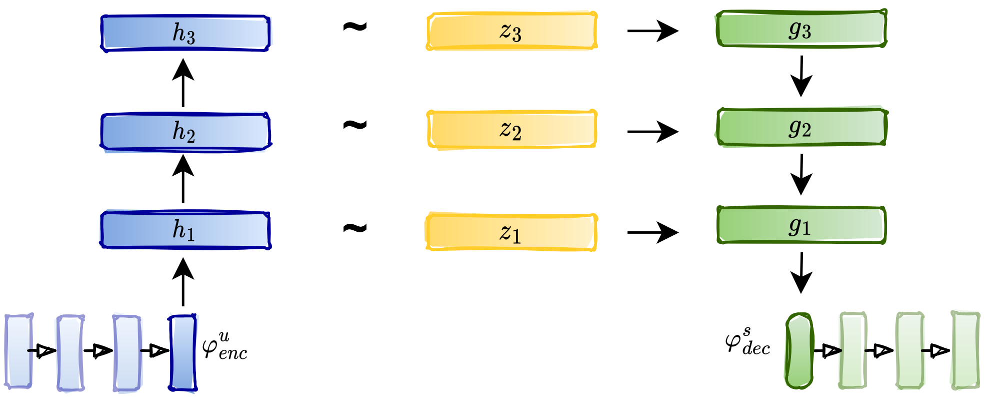


Figure 5.3: The architecture of our HVTOD model. The dialogue context is first processed with a recurrent network encoder to obtain a hidden representation. This representation then serves as an input to the hierarchical system of variational encoders stacked on each other. The latent variables are merged to form the decoder's initial hidden state.

encodes the context $C^t = (s_1, u_1, ..., s_t, u_t)$ with RNN-based utterance encoder to obtain a hidden representation of the context $h_0 = RNN_{\varphi^u_{enc}}(C^t)$. The obtained representation h_0 is subsequently used as input to a hierarchical L variational autoencoder system. The autoencoders are stacked on each other. Therefore, we refer to l-th VAE as VAE on level l or sometimes just *layer l*. Formally, each level l computes parameters of posterior distribution $q(z_l|C^t)$ to sample the latent variable as follows:

$$\begin{aligned} \mathbf{h}_l &= e_l(\mathbf{h}_{l-1}) \\ \mathbf{z}_l &\sim p(\mathbf{z}_l|C^t) = q_l(\mathbf{h}_l) \end{aligned} \tag{5.4}$$

where e_l, A_l are trainable matrices. In other words, each latent variable z_l on level l is sampled from a distribution conditioned on context and preceding VAE layers $q(z_l|C^t, h_1, ..., h_{l-1})$. We use Gumbel-softmax distribution to obtain discrete vectors similar to the previous section. To form the output, we aggregate the information on each layer in an output variable g_l as follows:

$$\mathbf{g}_l = D_l(z_l) + \alpha \cdot \mathbf{g}_{l+1}, \alpha_l \in [0, 1] \tag{5.5}$$

Where D_l is a trainable linear projection, and α_l is a scaling coefficient increased during training. All α_l are set to 1 in the trained network. The output from the bottom hidden layer g_1 generates the response s_{t+1} using an RNN decoder parametrized by φ^s_{dec}.

5.3.2 HVTOD training

In the LAVA framework, there are multiple stages of training. Here, we want to focus on unsupervised pre-training with an auto-encoding objective. However, we also evaluate the model performance in a supervised setting to understand the behavior better.

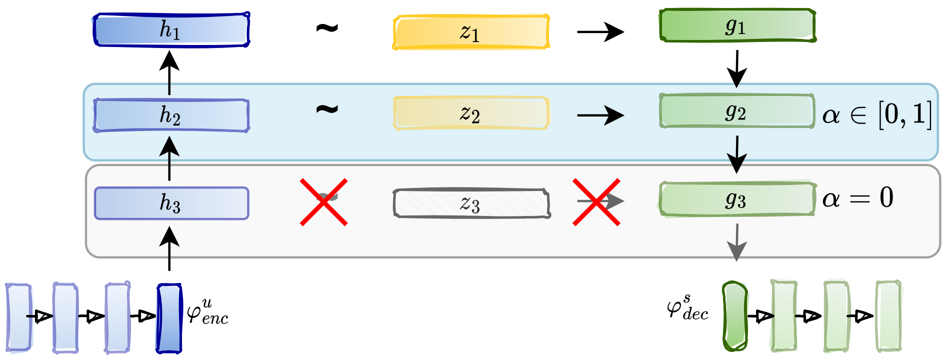


Figure 5.4: Our hierarchical variational model during training. At this stage, the top layer is used completely; the second layer is faded-in with increasing coefficient α while the bottom layer is not used yet, so it is effectively disconnected from the computation graph.

Training stages

We start the training with autoencoding. For the auto-encoding pre-training, we take an utterance x as input and follow the traditional approach to VAE training. We minimize the Evidence Lower Bound objective (ELBo), which we extend to reflect multiple levels of VAEs. The full minimalization objective, therefore, is:

$$\mathbb{L} = \mathbb{E}_{q(z_1|X^1),...,q(z_L|X^L)}[log\ p(x|z_1, ...z_L)] - \sum_{l=0}^{L} D_{KL}[q(z_l|X^l)||p(z)] \tag{5.6}$$

Where $X^L = x, z_1, ..., z_{L-1}$.

The second stage is fine-tuning the pre-trained model. We assume that the pre-training task helps to learn the network to create useful representations that can contribute to additional stages of training. To confirm this, we employ the second stage of training in which we use supervision in the form of belief state labels, which we include in the input. While in the autoencoding (AE) stage, the model learns to reconstruct the input utterance. During fine-tuning (FT), the model takes dialogue context and predicts a system response. FT stage utilizes the decoder pre-trained in the AE stage, and the encoder part is trained from scratch.

Training specifics

Training the network comprising multiple VAEs is challenging due to VAE training issues (Section 2.3). To simplify the training process, we incorporate the layers one by one. Only the top layer is initially trained; more layers are added later. To achieve this, we introduce a *fade-in coefficient* α (Li et al., 2020b) that is being annealed from $0 \rightarrow 1$ for each layer over a predetermined number of steps and serves to incorporate a new layer during training smoothly. Once the previous layer $l-1$ is trained, the new layer's output g_l is multiplied by the fade-in coefficient, which is initially less than one and increases over time to ensure a smoother transition when incorporating this layer. This process is depicted in Figure 5.4.

Furthermore, we experiment with an additional training objective – placeholder predictions. Since we work with delexicalized data, we can train the network to predict which placeholders are present in the utterance. We include this additional pseudo-supervision in the form of so-called *placeholder loss* $\mathbb{L}_{pl}$, which is computed simply as binary cross-entropy for each placeholder prediction.

5.3.3 Experiments

We are interested in evaluating two aspects of the proposed model. First, we explore how the multi-layer architecture contributes to the overall performance. Next, we evaluate the properties of learned latent variables. We use MultiWOZ 2.1 (See 2.6.1) for training and evaluation. We delexicalize (see Section 2.5.1) the utterances for training.

Unsupervised HVTOD

The second stage of HVTOD training, the fine-tuning, needs labels as supervision. We are interested in the task of dialogue response generation when no labels are available. Therefore, we also train an unsupervised variant of the model. We base this stage on the auto-encoding objective and employ the same mechanism as auto-encoding but train the model to predict the next response instead of reconstructing it.

Technical Details

For training the model, we use a single GPU, NVIDIA A30. We train the model for 100 epochs using the ADAM optimizer while we observe a decline in the validation loss. The learning rate is set to 10^{-3}. We use discrete latent variables of dimension 20 to represent latent dialogue actions. We determined the param-

no layers	placeholder loss	Success-unsup	BLEU-unsup
1	✗	0.20 ± 0.06	15.36 ± 0.90
1	✓	0.34 ± 0.06	16.43 ± 0.62
2	✗	0.32 ± 0.07	16.39 ± 0.67
2	✓	0.33 ± 0.07	16.31 ± 0.45
3	✗	0.35 ± 0.03	16.67 ± 0.48
3	✓	0.35 ± 0.02	16.92 ± 0.84

Table 5.8: The results of HVTOD architecture after fine-tuning on dialogue response generation in an unsupervised setting, i.e. without belief state labels as inputs. We provide means and standard error intervals computed over five runs.

eters by empirical experiments using grid search. To get a fair comparison, we adjust the number of latent variables used according to the number of layers in the model hierarchy. We adjust the number so that each configuration effectively has the same dimension of the latent space. In the supervised training stage, we represent the belief state as a one-hot discrete vector corresponding to specific slots.

5.3.4 HVTOD performance

We evaluate the performance of HVTOD with a different number of layers and with or without the placeholder loss for dialogue response generation in unsupervised (Table 5.8) and supervised (Table 5.9) setting. All variants are first trained with an autoencoding objective. We also evaluate the model quality after pre-training in Table 5.10.

Note that when we increase the number of layers, we decrease the dimension of the latent space accordingly. Therefore, all the compared models have the same number of parameters.

Autoencoding

We can see that in the autoencoding setting (Table 5.10), the increased number of layers helps to improve the performance concerning dialogue success and BLEU. Moreover, the dialogue success is also improved by including our additional placeholder loss.

Supervised fine-tuning The results of supervised fine-tuning are given in Table 5.9. This way, we achieve much higher scores, as expected. Also, we can see that the placeholder loss contributes to the overall model performance as does the number of used layers.

no layers	placeholder loss	Success-ft	BLEU-ft
1	✗	0.44 ± 0.08	17.37 ± 0.53
1	✓	0.52 ± 0.07	18.62 ± 0.52
2	✗	0.51 ± 0.03	19.17 ± 0.26
2	✓	0.55 ± 0.04	19.74 ± 0.36
3	✗	0.52 ± 0.04	19.36 ± 0.72
3	✓	0.55 ± 0.04	19.63 ± 0.41

Table 5.9: The results of HVTOD architecture after fine-tuning on dialogue response generation using belief state labels as inputs. We provide means and standard error intervals computed over five runs

Unsupervised response generation

For the unsupervised response generation (Table 5.8), we observe the benefits of using the placeholder loss when only one-level hierarchy is used. Surprisingly, we do not observe a similar effect when more layers are employed. Nevertheless, there are performance gains when using placeholder loss in the supervised setting (Table 5.9). This behavior shows that unsupervised dialogue response generation does not benefit from placeholder loss when more layers are employed. This suggests that one of the layers in a multi-layer hierarchy can capture information that can be gained from placeholder loss.

5.3.5 Latent space inspection

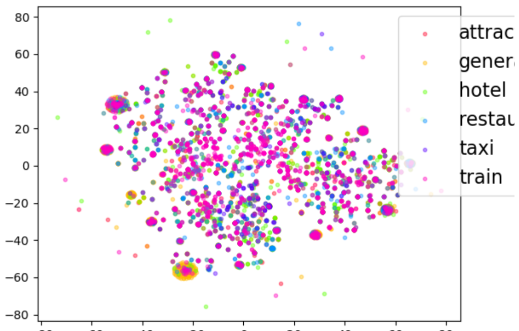
We inspect the structure of the learned latent space by transforming the variables using the t-SNE algorithm (Van der Maaten and Hinton, 2008) into 2-dimensional latent space and plotting them colored by their assignment to respective domains (Lubis et al., 2022). For autoencoding (Figure 5.5), the placeholder loss shapes the latent space to form more spread homogenous clusters, specifically on layer 1. The same influence is even more visible when we visualize latent spaces after training the model for dialogue response generation (Figure 5.6). Although the influence on latent space shape is clear, there is no significant difference in the final performance of the respective models. We also observe that although each layer contains useful information, there is some overlap between the layers. This suggests that it might be useful to enforce disentanglement of the latent variables somehow.

No. of layers	placeholder loss	Success	BLEU
1	✗	0.46 ± 0.02	55.39 ± 1.43
1	✓	0.58 ± 0.03	55.86 ± 1.17
2	✗	0.50 ± 0.02	59.07 ± 1.31
2	✓	0.62 ± 0.01	62.42 ± 1.16
3	✗	0.53 ± 0.02	58.84 ± 1.46
3	✓	**0.62** ± 0.01	**64.50** ± 0.96

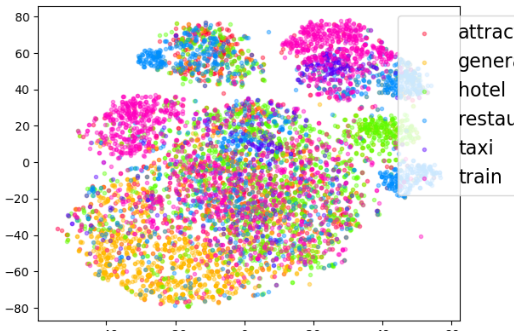
Table 5.10: The results of HVTOD architecture after pre-training with the unsupervised autoencoding objective. We provide means and standard error intervals computed over five runs. Note that dialogue success, in this case, indicates the reconstruction quality, but the model cannot be used for response dialogue generation since it's only trained on autoencoding.

5.3.6 Conclusion

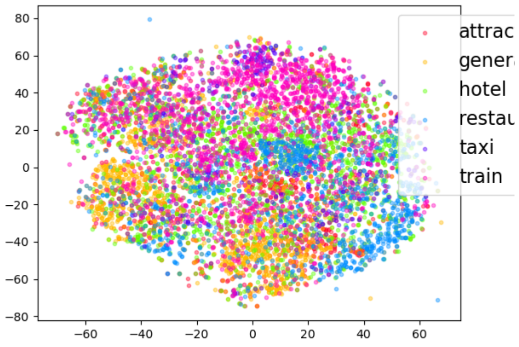
In this section, we introduced HVTOD, a hierarchical variational model for task-oriented dialogue. We wanted to assess the contributions of hierarchical structure to the overall model performance. Although the layer representations are not guaranteed to be distinct, we show that the model can leverage the benefits introduced by hierarchical structure since the variant with more layers outperforms the baseline (1-layer) variant with the same number of parameters. The visualizations of latent spaces also uncover that the learned representations are structurally different.



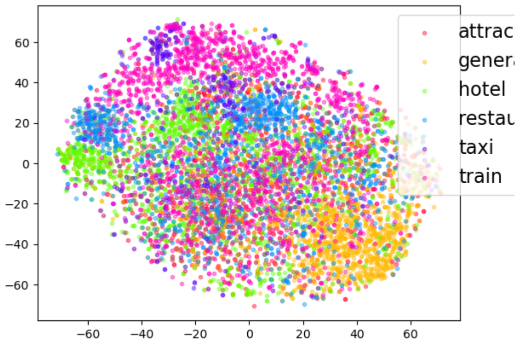
(a) level 1



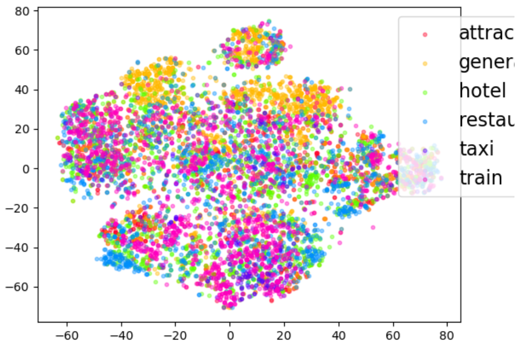
(b) level 1 + placeholder loss



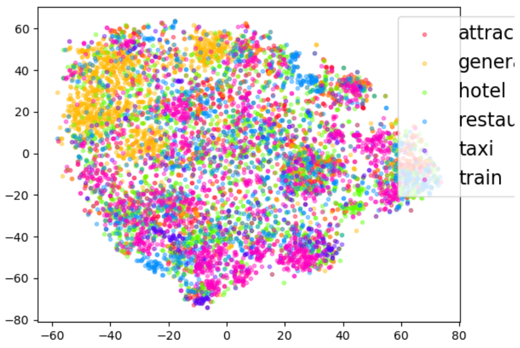
(c) level 2



(d) level 2 + placeholder loss

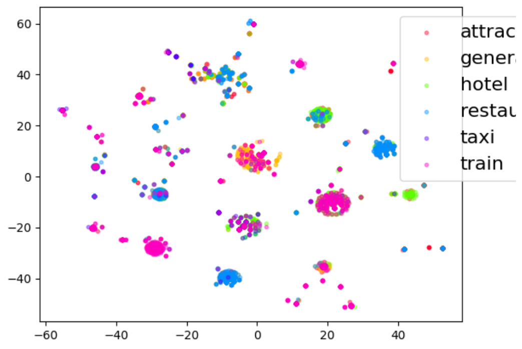


(e) level 3

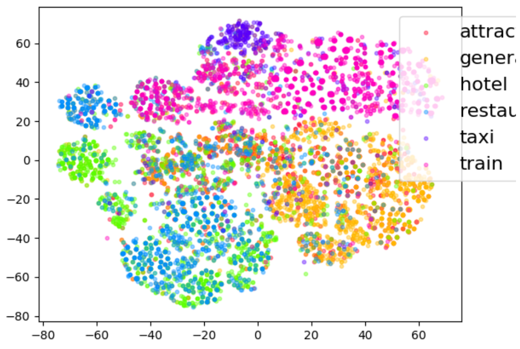


(f) level 3 + placeholder loss

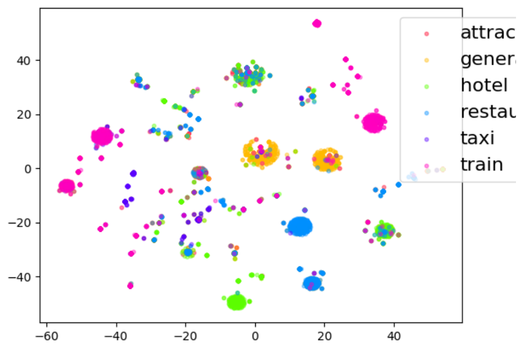
Figure 5.5: Visualization of t-SNE projections of latent variables after the autoencoding stage, sorted left-to-right. In the bottom row, we have a version using the additional placeholder loss. The colors correspond to respective domains.



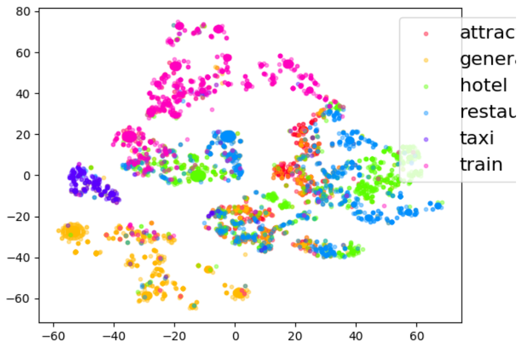
(a) level 1



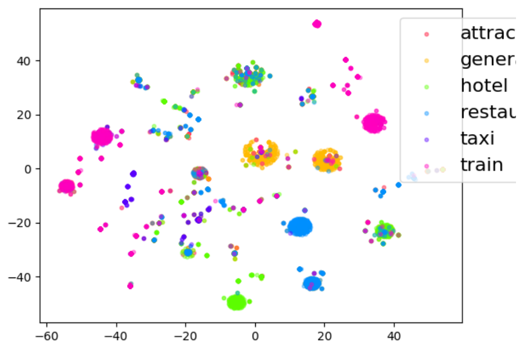
(b) level 1 + placeholder loss



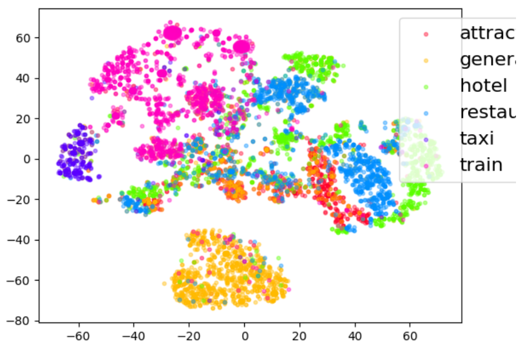
(c) level 2



(d) level 2 + placeholder loss



(e) level 3



(f) level 3 + placeholder loss

Figure 5.6: Visualization of t-SNE projections of latent variables after training for unsupervised dialogue response generation, sorted left-to-right. In the bottom row, we have a version using the additional placeholder loss. The colors correspond to respective domains.

5.4 Conclusion

In this chapter, we have presented the latent variable models trained with variational training methods. The TO-VRNN model (presented in Section 5.2.1) is based on recurrent architecture augmented with variational autoencoders. The HVTOD model (Section 5.3.1) encodes the whole dialogue context in each step and uses a hierarchical structure to model the latent space. Both models show promising performance, although the HVTOD model performs better overall. Moreover, we experiment with interpreting the representations formed in the latent space and observe that both models create representations that carry important and relevant information.

6 Sequence-to-Sequence Task-Oriented Dialogue Modeling

Using end-to-end trainable models instead of modular architectures (see Section 2.1) can can potentially offer more flexibility with respect to domain transfer, as only a single module needs to be adapted to new domains or use cases. In task-oriented dialogue, end-to-end implementations are dominated by sequence-to-sequence architectures based on language models (LM, Section 3.2). The LM-based approaches have taken over the benchmarks[1], demonstrating state-of-the-art performance.

However, these competitive models are fine-tuned on a large in-domain dataset, and domain transfer performance is not evaluated. In this chapter, we raise the question of how well these models can transfer the obtained skill of leading the dialogue to other domains. In other words, we want to discover if the models learn useful skills that can be beneficial in other domains or if the demonstrated behavior merely reproduces the patterns seen in the training portion of the data. We hypothesize that pre-training of these models can help to improve the performance. To confirm this hypothesis, we first describe our approach to end-to-end modeling with the GPT-2-based model (Kulhánek et al., 2021) in Section 6.1. We then describe our newly assembled and unified multi-domain dataset, designed for domain transfer experiments in Section 6.2 and detail our experimental results with AuGPT on this data in Section 6.3.

[1]`https://github.com/budzianowski/multiwoz#trophy-benchmarks`

The work presented in this chapter was published at the LREC conference and covered by Hudeček et al. (2022) [2]. For modeling, we use the AuGPT model (Kulhánek et al., 2021) to the development of which we contributed[3]. The published experiments in Hudeček et al. (2022) are further extended to determine if the model can learn from smaller training sets (6.3.3). We released the code for data preparation on GitHub[4].

6.1 AuGPT Model Description

We choose the AuGPT model introduced by Kulhánek et al. (2021). The architecture utilizes the GPT-2 model (Radford et al., 2019) for both belief state prediction and response generation, following the two-step method described in Section 2.5.3. GPT-2 is a pre-trained language model based on a Transformer decoder. Therefore, it is suitable for modeling sequences in natural language. When evaluated with automatic measures, the model performs well, and it showed promising behavior in human-judged interactive evaluation during the DSTC 9 (Gunasekara et al., 2020) challenge. AuGPT directly builds on the *small* version of the model, resulting in a 124M parameter model. Additionally, AuGPT introduces multiple training improvements. Instead of solely using cross-entropy loss for language modeling, AuGPT uses an additional training objective for state corruption detection. During training, the model is randomly presented with corrupted versions of the belief state (Peng et al., 2021b). For corruption, portions of the belief state are replaced with random values. Several strategies can be used to create the corrupted versions; AuGPT corrupts half of the samples by randomly applying one or more of the following modifications:

- Replace the belief state b with another belief state, sampled uniformly randomly from the training data.
- Replace the delexicalized response with a different randomly chosen one. If this change is combined with the first one, the delexicalized response and the belief state are taken from the same random sample.
- A different valid value is uniformly sampled for each slot in the belief state. In this case, the domain names and order are unchanged (i.e., the active domain is the same).

[2]This was a joint effort in which the author of this thesis focused on the data processing pipelines and AuGPT model training.

[3]The author of this thesis contributed to the model design and data preparation for the training

[4]`https://github.com/ufal/diaser`

The model learns to predict if the presented belief state corresponds to the respective context in the training example. This modification aims to improve the robustness of the belief state prediction. Since GPT can work with any natural language sentences, applying this model to our dialogue datasets is straightforward.

6.2 Diaser corpus introduction

Motivated by the questions raised in this chapter, we created a collection of several well-established task-oriented dialogue datasets spanning several domains to yield one larger multi-domain corpus which we call *Diaser*. Specifically, we used: **MultiWOZ 2.2** (MultiWOZ), **Schema-guided dialogue** (SGD), **DSTC2** (DSTC) and **CamRest676** (CamRest). For more details about the aforementioned datasets, please refer to Section 2.6. The merging process yields a dataset with over 37,000 dialogues, comprising more than 660,000 turns.

6.2.1 Theoretical motivation

We aim to cover as many domains as possible in a unified corpus. Our source dataset choice is thus mainly based on the level of annotation available – all source datasets include semantic annotation on the turn level and explicit database interaction. Despite the dataset similarities, important differences need to be resolved.

The main task when merging datasets is to unify the different domain-specific ontologies, i.e., the different sets of concepts included in the dialogue acts. More precisely, the unified dataset ontology contains all the possible domains with corresponding slots and associated value sets. We cannot consider the slots independently from the domain they belong to. Indeed, a slot that represents the *price range* will not have the same range of values when relating to a restaurant or a flight ticket. We identified two main problems related to this issue:

1. **Name reference ambiguities** We must design the final ontology so that different slot names refer to different concepts (with due precaution for label choice) and merge different slot names associated with the same value set under a single label. For example, in MultiWOZ, there are two different slot names *day* and *book-day* for the same value set (weekdays) and usage contexts. But in SGD, these slot names may be misleading since we can find a slot named *start-day* and another called *day*; the former refers to a calendar date while the latter refers to a weekday.

Data	SGD	MultiWOZ	DSTC	CamRest	Total
Domains	18	7	1	1	19*
Slots	145	29	10	7	166*
Dialogues**	22.8	10.4	3.2	0.7	37.1
Turns**	463.3	143.0	51.0	5.5	662.8
Turns/Dial.	20.30	13.71	15.77	8.12	17.83
Avg. utt. length	9.86	13.23	8.47	10.71	10.49
Unique Words**	32.3	23.2	1.3	1.7	49.9
Shannon ent.	8.96	8.54	7.04	7.69	9.01
Cond. ent.	4.76	4.41	2.14	2.95	4.83

Table 6.1: Composition of our dataset, with basic statistics, overall and for individual sources (number of domains and slots, total numbers of dialogues and turns, average number of turns per dialogue, and average utterance length in terms of words. * is not a sum due to ontology merging. ** in thousands.

2. **Absence of ontology/database** When the SGD dataset was collected, the authors used API calls instead of a database lookup. They also didn't provide an overall ontology. Therefore, no database-related metadata was released with the corpus, forcing us to create an ontology and a database for the data based on values occurring in the conversations.

6.2.2 Details of the merging approach

Here, we present details on how we merged the data into a common format, including handling different ontologies. Quantitative statistics of the final dataset are shown in Table 6.1. Full technical documentation can be found in the data repository.[5] Here we list an overview of all the required merging steps:

Matching belief representations. In DSTC and CamRest, belief state annotations are extracted from the user and the system utterance. In contrast, in the MultiWOZ and SGD datasets, the belief state is only extracted from the user utterance. We had to filter automatically the annotations from DSTC and CamRest until they matched the MultiWOZ belief state representation.

Adding meta features from the original datasets. DSTC2, CamRest, and MultiWOZ contain the *goal* of the dialogue (also called task) as a dialogue act with the constraints (e.g. expensive restaurant south) and the information the user needs to obtain from the system (e.g. phone number and address). They also contain a short text summarizing this goal for crowd workers. We include both versions of the task description with each dialogue.

[5]`https://github.com/ufal/diaser`

Intent	SGD	MultiWOZ	DSTC	CamRest	all
inform	151,467	84,259	33,451	5,786	274,963
request	81,786	31,888	13,620	1,516	128,810
offer	67,095	4,497	23,763	0	95,355
bye	35,089	12,380	0	0	47,469
else	31,030	13,779	0	0	44,809
select	33,820	2,834	243	0	36,897
confirm	30,327	0	5,756	0	36,083
thank	23,172	9,064	0	0	32,326
negate	132,01	4,399	4,413	0	22,023
total	435,957	163,100	81,346	7,312	718,645

Table 6.2: Most frequent intents in our unified schema, for each source dataset individually and overall (with absolute numbers of occurrences).

Unifying annotation structure. The final dataset structure is similar to the structure of SGD and MultiWOZ 2.2. We create a *Turn* object that contains either the user utterance and dialogue acts or the system utterance. Two consecutive entries for the user and system share the same turn number – we consider a *Turn* as an exchange between the user and the system (i.e. a pair or utterances).

Ontology unification. One of the most difficult parts of this work was unifying the ontologies of each original dataset because they were not built on the same dimensions. This concerns slot, domain, and intent names.

After indexing all annotations from the different datasets, we merged them manually using MultiWOZ as a golden reference to create the mapping. We always check the meaning in context and match slots/intents with the same semantics.

Slot co-reference Some source corpora (MultiWOZ and SGD) include co-reference between slots. For example, *start-time* can take the value “sooner than that” or the slot *hotel-name* can take the value “event you mentioned earlier”. This is a problem if we assume a self-contained ontology that captures all the possible values from the corpus. However, as these co-references are impossible to include in the ontology easily, we leave these values unchanged and the slot co-references are carried over to the unified data.

Slot	**SGD**	**MultiWOZ**	**DSTC**	**CamRest**	**all**
name	65,999	42,215	3,135	2	111,351
area	8,133	48,285	37,697	4,178	98,293
date	81,600	20,872	0	0	102,472
price	1,950	33,084	32,267	4,032	71,333
type	41,146	28,562	0	0	69,708
leave	32,625	26,684	0	0	59,309
arrive	26,180	26,228	0	0	52,408
food	13,501	20,963	3,007	571	38,042
book	17,618	11,723	0	0	29,341
total	298,752	232,388	76,106	8,783	532,257

Table 6.3: Most frequent slots in our unified schema, for each source dataset individually and overall (with absolute numbers of occurrences).

6.3 Experiments

In this chapter, we want to explore if sequence-to-sequence LM-based architectures can robustly learn the dialogue modeling capabilities and how well they can transfer them to other domains. To answer these questions, we conduct a series of experiments to train the model using only a small portion of the training data or even a different dataset with shared characteristics such as a task-oriented approach, regular user-system interaction, etc.

We divide the results into two sections. First, we describe the performance of the AuGPT model when trained and evaluated on various datasets. In the second section, we show the results when the model is pre-trained on a certain dataset and then fine-tuned using only a small portion of training data from the target dataset.

We are interested in the overall performance of the models. Therefore, we measure Joint Goal Accuracy and Slot-F1 and BLEU for response generation (see Section 2.7). We do not compute the dialogue success as it is not well-defined on all the sub-datasets.

6.3.1 Experimental setup

For training, we follow (Kulhánek et al., 2021) with the training setup – We use PyTorch framework (Paszke et al., 2019) and run the training for 8 epochs on the MultiWOZ data when all the training examples are used. The AuGPT model described in Section 6.1 consists of 12 transformer blocks with a model layer size

training			evaluation			metrics		
DSTC	**MW**	**SGD**	**DSTC**	**MW**	**SGD**	**Slot F1**	**JGA**	**BLEU**
✗	✓	✗	✗	✓	✗	0.89	0.53	18.61
✗	✗	✓	✗	✓	✗	0.16	0.02	4.01
✓	✓	✗	✗	✓	✗	0.89	0.55	19.67
✓	✗	✓	✗	✓	✗	0.17	0.03	5.68
✗	✓	✓	✗	✓	✗	0.89	0.52	19.92
✓	✓	✓	✗	✓	✗	0.90	0.54	21.09
✗	✓	✗	✗	✗	✓	0.04	0.01	5.63
✗	✗	✓	✗	✗	✓	0.59	0.21	28.17
✓	✓	✗	✗	✗	✓	0.03	0.01	5.51
✓	✗	✓	✗	✗	✓	0.58	0.21	27.96
✗	✓	✓	✗	✗	✓	0.63	0.23	27.54
✓	✓	✓	✗	✗	✓	0.63	0.22	27.72
✓	✓	✗	✓	✓	✓	0.28	0.12	15.30
✓	✗	✓	✓	✓	✓	0.55	0.22	27.28
✗	✓	✓	✓	✓	✓	0.65	0.25	25.13
✓	✓	✓	✓	✓	✓	0.70	0.28	**29.73**

Table 6.4: Performance of the AuGPT model trained and evaluated on various subsets of the unified dataset, namely DSTC, MultiWOZ (MW), and SGD. We omit CamRest since the data are very similar to DSTC.

equal to 768, having 124 million parameters in total. For the state corruption detection task, we use a dropout of 0.1 with label smoothing of 0.1. We use the AdamW optimizer (Loshchilov and Hutter, 2019) and employ mixed-precision training (Micikevicius et al., 2017).

For the data, we follow the dataset discussed in Section 6.2. We use different subsets for training and testing the model, as described for each experiment individually.

6.3.2 Influence of training data on the target performance

In Table 6.4, we can see a general pattern in the results, suggesting that the model fails to generalize across different datasets when a subset of data we evaluate is not included in the training. A significant drop in performance can be observed across all recorded metrics. We use DSTC, MultiWOZ (MW), and SGD. We omit CamRest since the data are very similar to DSTC.

Let us first observe some global properties. Regarding the F_1 slot score and joint goal accuracy, the explanation for the poor model's accuracy can be found in a substantially different distribution of slots across SGD/MultiWOZ datasets and DSTC. Some slots are present in just one of the datasets, others are mentioned in different contexts or with different frequencies. Training on MultiWOZ provides us with the highest scores on slot F_1 and joint goal accuracy. Training on both SGD and MultiWOZ brings only a very marginal improvement. The large difference in performance can also be seen in terms of BLEU, which suggests a vastly different language used in each dataset.

The performance of the model evaluated on MultiWOZ is analogous. Training the model solely on SGD gives us a model that cannot generalize to MultiWOZ data. Concatenating the MultiWOZ datasets with SGD/DSTC, or both during training, leads to a little improvement, predominantly in terms of BLEU score. We get the best results when using all three datasets for training and obtain the model with better generalizing capabilities supported by the BLEU score of 21.09.

The model evaluation on SGD data mostly follows the same pattern. However, in this case, it is quite interesting to observe that training the model on all three datasets is not beneficial, and the performance stagnates, although the training set is much larger.

Finally, if we evaluate all three datasets, it is clear that the best-performing model is obtained when we train it on the full data. Table 6.4 shows that including SGD data is crucial for achieving higher BLEU values while training on MultiWOZ helps the model predict slot values.

6.3.3 Domain adaptation with a small in-domain set

Table 6.5 presents results for domain adaptation experiments. We first train the AuGPT model on the MultiWOZ dataset and then use a small portion of the SGD data to adapt the model to this different dataset. The results show that although on MultiWOZ helps slightly to improve the final BLEU score, the performance of belief state tracking is not significantly improved, and the results are overall rather negative.

pre-training	fine-tuning	ft-ratio	**Slot-F1**	**JGA**	**BLEU**
–	SGD	100%	0.59	0.21	28.17
–	SGD	10%	0.47	0.15	20.13
MultiWOZ	SGD	10%	0.46	0.16	19.46
–	SGD	5%	0.29	0.08	14.65
MultiWOZ	SGD	5%	0.29	0.12	15.16
–	SGD	1%	0.07	0.03	6.37
MultiWOZ	SGD	1%	0.06	0.01	8.40
–	SGD	0.5%	0.04	0.04	3.43
MultiWOZ	SGD	0.5%	0.04	0.02	5.95

Table 6.5: Performance of the AuGPT model trained and evaluated on various subsets of the unified dataset.

6.3.4 Dynamics of Successful and Failed Dialogues

We also perform a detailed quantitative analysis of errors made by the trained models. We evaluated the AuGPT model trained on the MultiWOZ data with respect to the dialogue success rates. We only chose the MultiWOZ data due to the additional annotation available in MultiWOZ. Specifically, MultiWOZ also has user and system dialogues act annotated, which we use in this analysis. We consider dialogue success as defined in Section 2.7.

In Figure 6.1, we provide a histogram of dialogue state tracking errors in successful and failed dialogues. We further identify four reasons that can cause an error:

1. A *domain* is wrongly identified.
2. An *intent* is detected wrongly.
3. An *inform* value is captured incorrectly.
4. A *request* was not answered.

We go turn by turn and identify the errors in the dialogue state predicted by the system compared to the expected state. "All failed" and "all success" correspond to failed and successful dialogues. It is thus an evaluation concerning the dialogue as a whole, unlike the turn-level methods, such as evaluating the system's response generation. In particular, we show the count of cases where some information is understood wrongly by the system (positive values) or is missing completely (negative values). We can see that in successful dialogues, the system might make some mistakes at first but can recover eventually. On the other hand, most dialogue failures are caused by missing information in the second half of the dialogue, which suggests that the recovery is not certain.

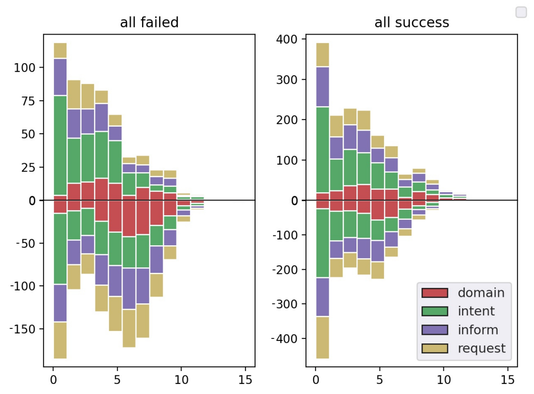


Figure 6.1: Distribution and types of dialogue failures that occurred with the GPT2-based model on the MultiWOZ data. The horizontal axis corresponds to turn numbers. Positive values represent cases where the information is captured wrongly, while negative values represent cases where particular information is missing. The source of error (incorrect domain or intent, missing information, or not providing a request) is depicted as color-coded.

We made some observations:

- For both successful and unsuccessful dialogues, the first turn concentrates the most errors.
- Generally speaking, the failed dialogues show much superfluous information in the first five speech turns, especially regarding the user's intent. On the other hand, from approximately the fourth speech turn onwards, there is a lot of missing information, especially at the level of the dialogue domain. If we add the missing information with the superfluous information, the error distribution peaks between turns three and seven of the dialogue.

6.4 Conclusion

In this work, we unified a large task-oriented dialogue corpora at both data and annotation levels, which requires a complex process of merging ontologies. We showed that additional data from other sources helps train LM-based end-to-end dialogue models when converted to a unified format in specific cases. However, it is not a general rule. Although the new dataset is still far from perfect coverage, it is a step towards wider and more authentic data. We also performed experiments to explore the ability of LM-based systems to adapt to new domains easily with a small number of in-domain examples. The negative results suggest that using in-domain data for domain adaptation is not straightforward.

7 Large Language Models for Task-Oriented Dialogue

As described in the previous chapter, pre-trained language models perform very well in end-to-end dialogue modeling. Despite this success, the widely used approach of fine-tuning pre-trained LM on a particular dataset still does not guarantee easy transferability of the learned knowledge, as we also show in Chapter 6. Large Language Models (LLMs) increased the model size by order of magnitude compared to the previous generation of pre-trained LMs. Undeniably, LLMs have transformed the NLP field, showing outstanding performance across many NLP benchmarks such as Winograd Challenge (Levesque et al., 2012) or GLUE (Wang et al., 2018). Instruction fine-tuning of LLMs can align the model outputs with human preferences (Ouyang et al., 2022; Wang et al., 2022) and improve the LLMs' communication capabilities substantially.

State-of-the-art LLMs are good at understanding user needs and can provide relevant answers. Consequently, we have seen many chatbot applications both inside and outside academia (ChatGPT[1], Claude[2] or Bard[3]) which build upon the raw power of instruction-fine-tuned LLMs.

In this chapter, we explore the abilities of LLM to model task-oriented dialogue using in-context learning. We introduce an LLM-based TOD conversation pipeline which resembles other approaches based on LMs (Peng et al., 2021a; Yang et al., 2021) in Section 7.2. The method uses state tracking and response generation as two main, separate steps while keeping the role of a dialogue policy

[1]`https://openai.com/blog/chatgpt`
[2]`https://www.anthropic.com/index/introducing-claude`
[3]`https://blog.google/technology/ai/bard-google-ai-search-updates/`

implicit (see also Sections 2.5.3, 6.1). However, instead of fine-tuning LMs, it intentionally relies almost exclusively on using pre-trained LLMs as-is so we can test their out-of-the-box capabilities. The dialogue context and domain description are introduced to the model only by including them in the input prompt. We experiment with zero-shot and few-shot approaches (see Section 2.2.1) and describe the experiments in Section 7.3. In the zero-shot setting, the model receives a domain description only; in the few-shot setting, it additionally uses a few retrieved examples (see Section 7.2.1 for details). We evaluate the performance regarding response generation and dialogue success and discuss the results in Section 7.4. To obtain more insights, we also perform a human evaluation to test more real-world scenarios and present the analysis in Section 7.5.

This work was published at the SIGDial 2023 conference (Hudeček and Dušek, 2023) and was extended in this chapter with more details and additional experiments. Our experimental code is available on GitHub[4].

7.1 Motivation

Given the millions of daily interactions with LLM-based chatbots, these models can handle users' needs satisfactorily, at least to some extent. However, these chatbots are tuned using unstructured open-domain conversations. We aim to evaluate these systems on task-oriented dialogues, where the system has to follow a predetermined structure and handle external sources of information, such as APIs or databases. We ask to what extent LLMs can handle these applications off-the-shelf, i.e., without fine-tuning. This approach, frequently referred to as *in-context learning* or *prompting*, is a common way to work with LLMs and offers competitive performance. Moreover, TOD systems output in-domain information with a predetermined structure and lend itself well to evaluation, thanks to pre-existing annotated data sets. We avoid fine-tuning techniques and focus on zero-shot or few-shot settings using in-context learning, as this approach has lower hardware requirements and barrier of entry and better flexibility or even performance in certain tasks (Su et al., 2022). It is important to note that we cannot exclude the possibility that some models were exposed to our selected datasets during training (Golchin and Surdeanu, 2023). However, evaluating this setting is important as the real-world use cases might largely rely on this approach.

[4]`https://github.com/vojtsek/to-llm-bot`

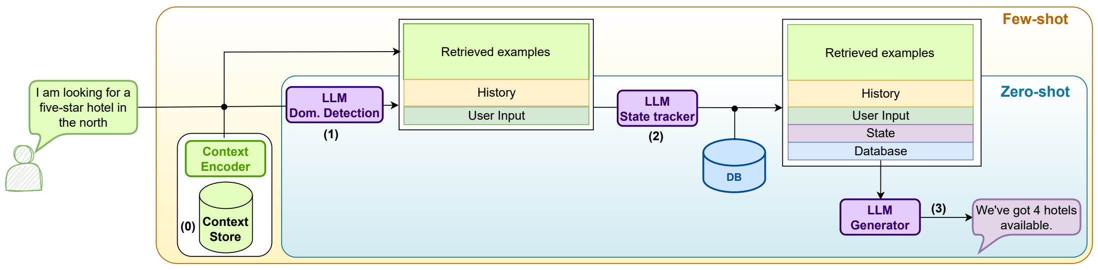


Figure 7.1: A detailed description of our proposed pipeline. (0) As a preprocessing step, we encode a subset of the training set that will be used to retrieve few-shot examples. Given a user input, we: (1) Detect the domain, retrieve relevant examples (in the few-shot setting), and construct an initial prompt. (2) Infer the belief state using LLM. Based on that, we retrieve database information and construct another prompt that includes the state and database results. (3) We ask the LLM to provide a final response.

Our experiments show that LLMs are not very good at state tracking, and their performance falls behind state-of-the-art, task-specific trackers. However, if provided with correct belief states, some yield interesting response generation performance comparable to earlier fine-tuned state-of-the-art models. Moreover, our human evaluation experiments show that LLMs are generally good with human interactions, and their performance cannot be assessed only based on automatic evaluations.

7.2 Method

An overall description of the proposed pipeline is shown in Figure 7.1. The system consists of a pre-trained LLM and an (optional) context store in a vector database. Three LLM calls are performed in each dialogue turn, with specific prompts (see Section 7.2.1). First, the LLM performs domain detection and state tracking (Section 7.2.2). The updated belief state informs a database query, the results of which are used in the subsequent LLM-based response generation step 7.2.4. In the few-shot setting, the context store stores a limited number of examples from the training set, which are retrieved based on similarity with the conversation context and included in LLM prompts (see Section 7.2.3).

Prompt	`Determine which domain is considered in the following`
	`dialogue situation.`
	` Choose exactly one domain from this list:`
	` restaurant, hotel, attraction, taxi, train`
	` Answer with only one word, the selected domain from the list.`
	` You have to always select the most probable domain.`
	`------- Example 1: --------`
	`Customer: I need a cheap place to eat`
	` Assistant: We have several not expensive places available.`
	` What food are you interested in?`
	`Customer: Chinese food.`
	`Domain: restaurant`
	`------ Example 2: --------`
	`Customer: What is the address?`
	`Assistant: It's 123 Northfolk Road.`
	` Customer: That's all. I also need a train from London.`
	`Domain: train`
	`-----------`
	`Now complete the following example:`
	`Customer: I am looking for a cheap place to stay.`
	`Domain:`

Output:	`hotel`

Table 7.1: A prompt used for domain detection for MultiWOZ. It contains task definition, domains description, static examples and user utterance, coded by respective colors.

7.2.1 Prompt construction

We aim to compare the raw capabilities of the selected LLMs. Therefore, we do not focus on prompt engineering techniques and choose universal prompts for all LLMs in this work. We choose simple, plain language statements as prompts, with no specific vocabulary, based only on a few preliminary tests. We define a single **domain detection prompt** (Table 7.1) for all examples, plus a pair of prompts for each domain in the given dataset: a **state tracking prompt** (Table 7.2) and a **response prompt** (Table 7.3).

The domain detection prompt includes a task description and two static examples of domain detection. In addition to general instructions, each state tracking prompt contains a domain description, a list of relevant slots, the dialogue history, and the current user utterance. The response prompts do not contain the per-domain slot list but include the current belief state and database results. In the few-shot setting, each tracking and response prompt contains positive and negative examples retrieved from the context store (see Section 7.2.3).

Prompt	`Definition: Capture entity values from last utterance`
	`of the conversation according to examples.`
	`Capture pair "entity:value" separated by colon and no spaces`
	`in between. Separate entity:value pairs by hyphens.`
	`If not specified, leave the value empty.`
	`Values that should be captured are:`
	`- "pricerange": the price of the hotel`
	`- "area" that specifies the area where the hotel is located`
	`(north/east/west/south/centre)`
	`- "internet" that specifies if the hotel has internet (yes/no)`
	`- "parking" that specifies if the hotel has parking (yes/no)`
	`- "stars" that specifies the quality of the hotel (1/2/3/4/5)`
	`- "type" that specifies the type of the hotel`
	`(hotel/bed and breakfast/guest house)`
	`[history]`
	`Customer: "I want a cheap place to stay."`

Output:	`pricerange:"cheap"`

Table 7.2: A zero-shot version of the prompt used for state update prediction for MultiWOZ 2.2. It contains task definition, domain description, dialogue history and user utterance, coded by respective colors.

Prompt	`Definition: You are an assistant that helps people`
	`to book a hotel.`
	`The user can ask for a hotel by name, area, parking,`
	`internet availability, or price.`
	`There is also a number of hotel in the database currently`
	`corresponding to the user's request.`
	`If you find a hotel, provide [hotel_name], [hotel_address],`
	`[hotel_phone] or [hotel_postcode]`
	`Do not provide real entities in the response! Just provide`
	`entity name in brackets, like [name] or [address].`
	`If booking, provide [reference] in the answer.`
	`[history]`
	`Customer: "I want a cheap place to stay."`
	`State: hotel { pricerange: "cheap"}`
	`Database: hotels: 23`

Output:	`We have 23 [pricerange] hotels available,`
	`do you have a location preference?`

Table 7.3: A zero-shot version of the prompt for response prediction for MultiWOZ 2.2. It contains task definition, domain description, dialogue history, user utterance and belief state with database results, coded by respective colors

7.2.2 Domain Detection and State Tracking

We prompt the LM twice at each turn during state tracking, first to detect the active domain and then to output a belief state update. We then use the outputs to update the accumulated global belief state.

The two prompting steps are used since we need the models to operate in a multi-domain setting, i.e., handle conversations spanning multiple domains. Therefore, we need to be able to detect the current active domain. We achieve this by first prompting the LLM with a domain detection prompt (using a single prompt for all examples). This prompt (see Table 7.1) is static, i.e., it remains the same for each example.

Once we obtain the active domain prediction, we can include manually designed domain descriptions in a second prompt that handles belief state prediction. We ask the model to predict values that changed or appeared in the current turn. An example of a prompt used for state tracking is provided in Table 7.2. For the few-shot variants, we retrieve few-shot examples from the context store (Section 7.2.3), limited to the active domain. For this purpose, each conversation snippet contained in the context store comes from a single-domain conversation.

Our preliminary experiments showed that LLMs consistently struggle to output all active slot values at every turn. Therefore, we model only state updates, following the MinTL approach (Lin et al., 2020). Here, the model only generates the slot-value pairs that have changed in the current turn. The global belief state is then accumulated using these turn-level updates. To obtain machine-readable outputs useful for database queries or API calls, we specify in the prompt that the model should provide JSON outputs, and any provided few-shot examples are formatted accordingly. The current belief state is used to query the database for entries matching all user-specified slots in the active domain. Given the belief state and database results, the response generation is straightforward. The prompt for the LLM includes dialogue history, user utterance, belief state, and database results (and retrieved examples in the few-shot setting). It requests the model to provide a fitting system response. We generate delexicalized responses (see Section 2.5.1).

In addition to simplifying the task for the model, delexicalized outputs allow us to evaluate the success rate and compare it to previous works. The prompt specifies that the model should provide entity values as delexicalized placeholders, and any few-shot examples are constructed accordingly.

7.2.3 Context Storage

It has been shown that enriching prompts with specific examples (i.e. *few-shot prompting*) boosts LM performance (Madotto et al., 2020; Brown et al., 2020). To apply this knowledge efficiently in our pipeline, we introduce a storage that contains encoded dialogue contexts. This context storage is optional and is only required for the few-shot prompting variant. We use dialogue context taken from a fixed-length history window as the key to be encoded in the vector database. Once the relevant examples are retrieved, we include them in the prompt to guide the model better. Some LLMs rely on negative (counter-) examples as well (Wang et al., 2022). Therefore, we follow Peng et al. (2021a)'s consistency classification task approach to produce negative examples: We take some of the retrieved belief state examples, corrupt them by replacing some of the correct slot values with random values, and present them as negative in the prompt for the Tk-Instruct model. When constructing the context store, we employ only a few training examples, ensuring we evaluate in a truly few-shot setting.

7.2.4 Response Generation

The prompt used for the final response generation in a zero-shot version is depicted in 7.3. In the few-shot version, examples are also included. The model is instructed to provide delexicalized responses (see Section 2.5.1) to be able to place correct values from database results. Also, the standardized evaluation scripts work with delexicalized utterances.

7.3 Experimental Setup

To obtain a broad overview of the current LLMs' capabilities, we compare several models spanning different numbers of trainable parameters and different training methods. We also experiment with four variants of the base setup, using either zero-shot or few-shot operations and either predicted or oracle belief states.

7.3.1 Tested Models

We chose the following five instruction-tuned models for our experiments, spanning different sizes (within the limitations of hardware available) and using freely available models and the paid ChatGPT API. We directly indicate the model name's specific model variant (i.e., model size, given by the number of parameters).

- **Tk-Instruct-11B**[5] (Wang et al., 2022) is based on the T5 encoder-decoder architecture (Raffel et al., 2020a). It was tuned on a dataset of over 5M task instances with instructions.

- **ChatGPT** is a product introduced by OpenAI.[6] Although the exact training process and architectures were not published, it most probably uses a similar architecture and fine-tuning techniques as InstructGPT (Ouyang et al., 2022), with additional human feedback. We use the *gpt-3.5-turbo* model and API version `0301`.

- **Alpaca-LoRA-7B** [7] is a version of the LLaMa model (Touvron et al., 2023) using the LoRA method (Hu et al., 2021) for fine-tuning on the Stanford Alpaca project data (Taori et al., 2023). LoRa keeps the base model parameters frozen but adds smaller weight matrices to transform its outputs.

- **GPT-NeoXT-Chat-Base-20B**[8] is based on the GPT-NeoX open-source language model (Black et al., 2022) and fine-tuned with over 40M dialogue-style instructions.

- **OPT-IML-30B**[9] (Iyer et al., 2022) is based on the Transformer decoder OPT model (Zhang et al., 2022) and trained with a custom set of instructions, including the fine-tuning set from the Tk-Instruct model.

7.3.2 Evaluated variants

We test four setup variants for each pair of model and dataset. Specifically, we use zero-shot (without examples) or few-shot (including examples) prompts (*-zs-* vs. *-fs-*) and either generated or oracle belief states (*-gbs* vs. *-obs*). For retrieval in the few-shot setting, we store just 10 examples per domain in the context store by default. We experiment with increasing this number in Section 7.4.4. Using the oracle belief state allows us to focus on evaluating the LLM's ability to guide the dialogue.

[5]`https://huggingface.co/allenai/tk-instruct-11b-def-pos-neg-expl`
[6]`https://openai.com/blog/chatgpt`
[7]`https://huggingface.co/tloen/alpaca-lora-7b`
[8]`https://huggingface.co/EleutherAI/gpt-neox-20b`
[9]`https://huggingface.co/facebook/opt-iml-30b`

7.3.3 Experiment Details

Due to the expensiveness of the LLM runs (hardware requirements for the freely available models and actual cost for ChatGPT), we did not perform a grid search but used a limited set of preliminary experiments to determine hyperparameters. Based on this, we used the context of two preceding utterances (one user + one system) as the context store keys (cf. Section 7.2.3). We retrieve two examples for few-shot prompts and make one corrupted variant for negative examples. To corrupt an example, we randomly switch some of the slot values, similarly to Kulhánek et al. (2021), Section 6.1. In the context store, we encode few-shot examples using the multilingual embedding model provided by Reimers and Gurevych (2020)[10] and store them in the FAISS database (Johnson et al., 2019). To perform the LLM calls, we use the Huggingface library[11] and the OpenAI API.[12]

7.3.4 Evaluation Measures

We evaluate the system outputs on multiple levels using automatic metrics and human evaluation. Results are given in Sections 7.4 and 7.5, respectively.

Automatic Metrics

We first follow the LLM calls in automatic evaluation and evaluate domain detection, state tracking, and response generation. We also evaluate the overall dialogue-level performance. Please see Section 2.7 for details about the used metrics.

For *domain detection*, we compute **detection accuracy**. For *state tracking*, we compute **micro-F1** score and **Joint Goal Accuracy** (JGA). To evaluate *response generation*, we follow related works and use **BLEU score**. The main *overall measure* for evaluating a task-oriented dialogue is the dialogue **success rate** (Deriu et al., 2021).

[10] `https://huggingface.co/sentence-transformers/all-mpnet-base-v2`
[11] `https://huggingface.co`
[12] `https://platform.openai.com`

Human Evaluation

For human evaluation, we perform a small-scale in-house interaction study on MultiWOZ. The annotators were given goal descriptions sampled from the MultiWOZ test set and concise instructions on how to proceed. Since the MultiWOZ goal often involves tasks in multiple domains, we ask annotators to evaluate each domain in the dialogue distinctly. At the end of each dialogue, the annotators are asked to answer these questions:

1. *How many of the sub dialogues/domains were handled successfully?* (corresponding to dialogue success)

2. *How many clarifications or corrections were needed?*

3. *Was all the provided information captured correctly?* (corresponding to JGA)

More details on the annotation process and instructions are given in Section 7.5.1

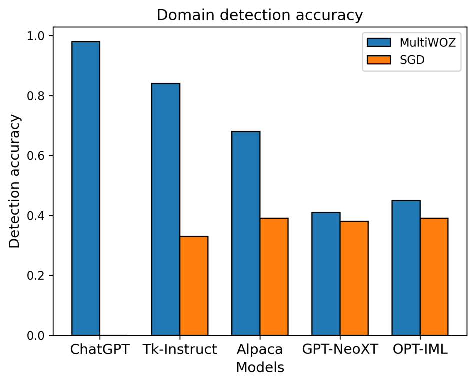


Figure 7.2: Domain detection accuracy concerning different models for MultiWOZ 2.2 and SGD data, which consist of 7 and 18 domains, respectively.

7.4 Automatic Metrics Results

7.4.1 Domain detection

We report the domain detection accuracy on MultiWOZ and SGD in Figure 7.2. We observe that the domain detection accuracy varies quite a lot for most models. This presumably affects the quality of the retrieved few-shot examples and the appropriateness of the subsequent prompts. However, it is important to note that domain detection is turn-based, and arguably, some situations (e.g., providing an address, saying goodbye, etc.) are always handled similarly, even though they belong to different domains. Therefore, not all the retrieved examples from misclassified domains necessarily contain unrelated contexts. To explore this, we measure the performance of all models in case an oracle domain is given to them (Figure 7.3). Interestingly, using the Oracle domain did not improve performance; it even worsened in some cases. This suggests that the model-predicted domain is generally good enough, and additionally, providing the domain information does not contribute to the final system performance. The negative influence on performance might be caused by forcing the system to filter out relevant examples. The conversation snippets are domain-independent in multiple cases so that the retrieval might perform better even with a wrongly selected domain. Forcing the ground truth domain examples in these cases can be potentially harmful.

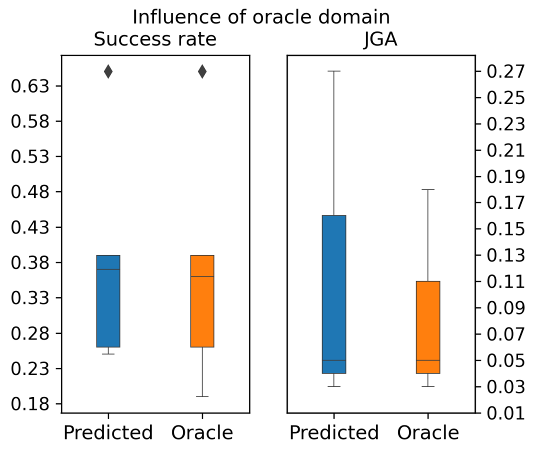


Figure 7.3: The influence of using oracle domain to retrieve examples. Interestingly, the oracle domain does not improve the performance, suggesting that the model-based detection is good enough for retrieval.

7.4.2 Response Generation

BLEU scores are low overall, far below the supervised state-of-the-art. Tk-Instruct and ChatGPT are the strongest here and perform roughly on par. This behavior is likely because the models were not fine-tuned on the particular datasets, therefore, they do not resemble the wording and phrasing used in this data. Nevertheless, this observation does not necessarily imply that the generated responses are incorrect.

model	few shot	oracle BS	MultiWOZ 2.2 BLEU	JGA	Slot-F1	Success
Supervised SotA	✗	✗	19.90♣	0.60♢	–	0.82♡
Alpaca-LoRA-7B-*zs-gbs*	✗	✗	1.61	0.06	0.07	0.04
Tk-Instruct-11B-*zs-gbs*	✗	✗	2.48	0.04	0.04	0.04
GPT-NeoXT-20B-*zs-gbs*	✗	✗	0.52	0.03	0.02	0.04
OPT-IML-30B-*zs-gbs*	✗	✗	0.56	0.02	0.04	0.03
ChatGPT-*zs-gbs*	✗	✗	4.17	0.13	0.40	0.31
Alpaca-LoRA-7B-*zs-obs*	✗	✓	1.73	–	–	0.08
Tk-Instruct-11B-*zs-obs*	✗	✓	2.66	–	–	0.18
GPT-NeoXT-20B-*zs-obs*	✗	✓	0.60	–	–	0.06
OPT-IML-30B-*zs-obs*	✗	✓	0.54	–	–	0.06
ChatGPT-*zs-obs*	✗	✓	3.76	–	–	0.47
Alpaca-LoRA-7B-*fs-gbs*	✓	✗	5.53	0.06	0.08	0.06
Tk-Instruct-11B-*fs-gbs*	✓	✗	6.56	0.16	0.33	0.19
GPT-NeoXT-20B-*fs-gbs*	✓	✗	2.73	0.05	0.04	0.05
OPT-IML-30B-*fs-gbs*	✓	✗	4.40	0.03	0.03	0.04
ChatGPT-*fs-gbs*	✓	✗	6.77	**0.27**	**0.51**	0.44
Alpaca-LoRA-7B-*fs-obs*	✓	✓	5.96	–	–	0.41
Tk-Instruct-11B-*fs-obs*	✓	✓	**6.91**	–	–	0.46
GPT-NeoXT-20B-*fs-obs*	✓	✓	2.92	–	–	0.28
OPT-IML-30B-*fs-obs*	✓	✓	5.40	–	–	0.28
ChatGPT-*fs-obs*	✓	✓	6.84	–	–	**0.68**

Table 7.4: Evaluation of the chosen LLMs concerning widely used TOD measures on the MultiWOZ dataset. For each model, we provide multiple variants. We use either zero-shot or few-shot prompts (-*zs*- vs. -*fs*-) and either generated or oracle belief state (-*gbs* vs. -*obs*). The few-shot variants use 10 examples per domain in the context storage, two selected for the prompts. We also provide supervised state-of-the-art results to put the numbers in context: ♣Sun et al. (2022), ♢Huang et al. (2023), ♡Feng et al. (2023).

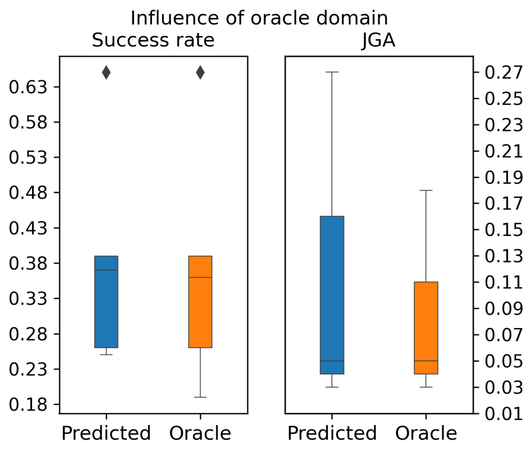


Figure 7.4: The influence of using oracle domain to retrieve examples. Interestingly, the oracle domain does not improve the performance, suggesting that the model-based detection is good enough for retrieval.

7.4.3 Belief State Tracking

The belief state tracking results overview is given in Tables 7.4 and 7.5 (*JGA* and *Slot-F1*). A huge gap exists between the state-of-the-art supervised fine-tuned models' performance and the LLM results. Also, our instruction-tuned LLMs fall short compared to Hu et al. (2022a), who used few-shot in-context learning to formulate state tracking prompts for big LLMs such as OpenAI `davinci-codex` (Chen et al., 2021) with 175B parameters and reported JGA 43.13% with a comparable number of examples used for few-shot retrieval. However, our models are generally an order of magnitude smaller, and we also use fewer examples in the prompt. We hypothesize that the performance could be further improved by careful model-specific prompt customization and perhaps task re-formulation; nevertheless, this is not the goal of this work. We intentionally focus on the universal framing of the task since we want to explore the general ability of the models to follow instructions.

When comparing the results among the models, ChatGPT outperforms the rest of the models by a large margin. Interestingly, the few-shot vs. zero-shot setting does not influence the results much for tracking, except for the GPT-NeoXT model.

model	few shot	oracle BS	**Schema Guided Dialogues** BLEU	JGA	Slot-F1	Success
Supervised SotA	✗	✗	29.90*	0.30†	0.60*	–
Alpaca-LoRA-7B-*zs-gbs*	✗	✗	2.79	0.02	0.01	0.11
Tk-Instruct-11B-*zs-gbs*	✗	✗	4.16	0.05	0.03	0.10
GPT-NeoXT-20B-*zs-gbs*	✗	✗	0.45	0.01	0.01	0.17
OPT-IML-30B-*zs-gbs*	✗	✗	1.63	0.01	0.01	0.17
Alpaca-LoRA-7B-*zs-obs*	✗	✓	2.76	–	–	0.23
Tk-Instruct-11B-*zs-obs*	✗	✓	5.21	–	–	0.24
GPT-NeoXT-20B-*zs-obs*	✗	✓	0.83	–	–	0.22
OPT-IML-30B-*zs-obs*	✗	✓	1.94	–	–	0.22
Alpaca-LoRA-7B-*fs-gbs*	✓	✗	6.32	0.04	0.01	0.09
Tk-Instruct-11B-*fs-gbs*	✓	✗	6.66	0.06	0.05	0.10
GPT-NeoXT-20B-*fs-gbs*	✓	✗	1.62	0.04	0.02	0.09
OPT-IML-30B-*fs-gbs*	✓	✗	0.82	**0.06**	**0.07**	0.08
Alpaca-LoRA-7B-*fs-obs*	✓	✓	6.99	–	–	**0.25**
Tk-Instruct-11B-*fs-obs*	✓	✓	**8.56**	–	–	**0.25**
GPT-NeoXT-20B-*fs-obs*	✓	✓	1.97	–	–	0.24
OPT-IML-30B-*fs-obs*	✓	✓	0.56	–	–	0.22

Table 7.5: Evaluation of the chosen LLMs concerning widely used TOD measures on the SGD dataset. For each model, we provide multiple variants, as described in Table 7.4. We also provide supervised state-of-the-art results to put the numbers in context: *Zhu et al. (2022), †Feng et al. (2021).

7.4.4 Dialogue-level performance

Results for dialogue success are provided in Tables 7.4 and 7.5, and there is again a large gap between prompted LLMs and supervised custom models' performance. ChatGPT seems to outperform other models, similarly to state tracking (cf. Section 7.4.3). In most cases, adding the retrieved few-shot examples helps. The contribution of retrieved examples is more obvious when we supply the oracle belief state, which helps consistently for all the models.

We also explore the influence of the context storage size on the dialogue success rate. The results are given in Figure 7.5. The biggest improvement can be achieved by supplying just a few examples instead of zero-shot prompting, but increasing the size of the example pool for retrieval does not yield further performance gains.

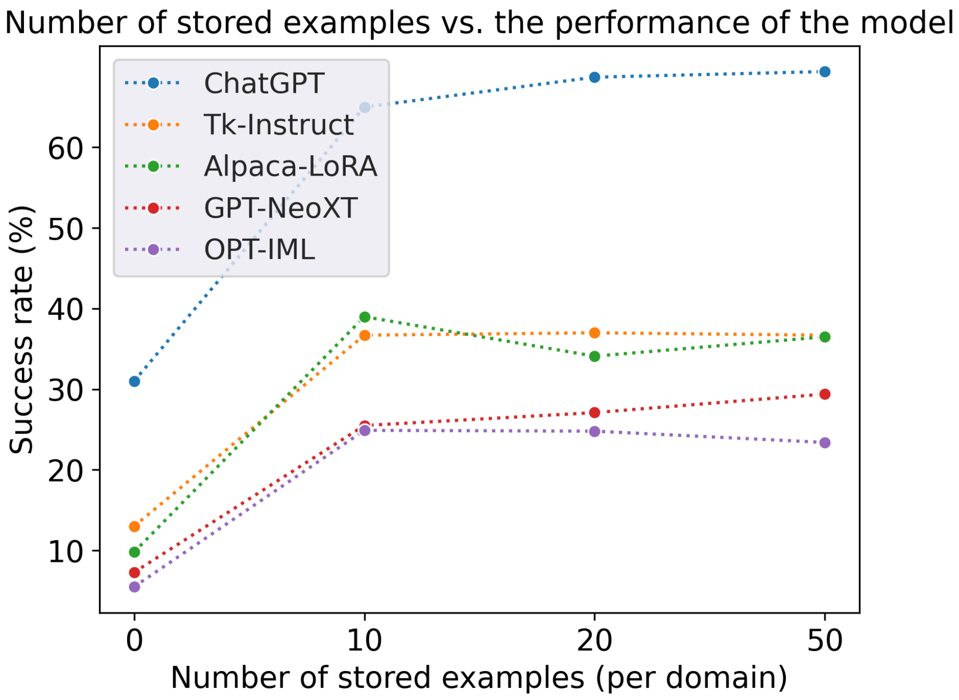


Figure 7.5: The influence of the number of examples per domain available for few-shot retrieval and response generation performance of the model in terms of the dialogue success on MultiWOZ 2.2 data with the oracle state supplied. Note that this does not represent the number of examples selected for the prompt, which is fixed to two.

domain	**Attraction**	**Taxi**	**Train**	**Hotel**	**Restaurant**
success	0.65	0.81	0.86	0.64	0.82

Table 7.6: Dialogue success of ChatGPT model when restricted to single domain dialogues in MultiWoz.

7.4.5 Single-domain performance

We evaluated the performance of the ChatGPT model when restricted to single-domain dialogues on MultiWOZ in Table 7.6. The average success rate is 0.76, which performs better then in case of the automatic metrics. This outcome is expected as single-domain dialogues are typically shorter and easier to manage. However, this observation can also give us interesting insights into the different performance values on the individual domains, which does not necessarily correlate with the number of slots or average dialogue length.

7.5 Model Analysis

7.5.1 Human Evaluation

We employed 6 annotators with a background in linguistics and NLP and let them interact with the two strongest models in terms of automatic metrics: ChatGPT and Tk-Instruct. We present the results in Table 7.8. We can see that in a real interaction with a human user and allowing for clarification or correction, the models perform better than the strict automatic evaluation. Furthermore, the models are often successful in multiple sub-dialogues, even if a part of the dialogue fails. The experiment also confirms the superior performance of ChatGPT on dialogue success and JGA. Not surprisingly, given the above results, conversations with ChatGPT also required fewer clarification turns than with Tk-Instruct.

7.5.2 Error Analysis

To better understand the models' behavior, we manually inspect a random sample of ca. 20 dialogues for each model, chosen from cases where the automatic success metric was not satisfied. In general, we can split most erroneous behaviors into two distinct groups, which we call *prompt-recoverable* and *inherent.*

Goal	You are looking for a train. The train should depart from Cambridge and should leave after 9pm on thursday and should go to Stansted Airport. Once you find the train you want to make a booking for 2 people Make sure you get the reference number
Customer	i need to find a train from cambridge to stansted airport
Assistant	what day will you be leaving?
Customer	i need to leave on thursday after 9pm.
Assistant	there is one on Thursday at 21:08, would you like me to book it for you?
Customer	yes please, for 2 people
Assistant	Okay a honda will pick you up, the contact is [phone]. Do you need anything else?
Customer	i need to book the train for 2 people
Assistant	a honda will pick you up from cambridge and the contact number is [phone].
Customer	i need a reference number for the train
Assistant	TR8383 leaves Cambridge at 21:08 and arrives at Stansted Airport at 21:36.

Table 7.7: A sample evaluation conversation between human and LLM-based (ChatGPT) agent. The sample includes a textual goal description that instructs the user. The slot values are lexicalized in the assistant's responses. Here, the system fails to provide the requested information due to irrelevant examples (from taxi domain) in the retrieved context. The slot values are highlighted for both informed and requested values.

	ChatGPT	Tk-Instruct
dialogues	25	25
subdialogues	52	48
clarify / dial	1.08	1.68
succesful subdialogues	81%	71%
succesful dialogues	76%	64%
correctly captured	88%	66%

Table 7.8: Human evaluation results for ChatGPT and Tk-Instruct-11B models. We evaluate the conversation on the sub-dialogue level i.e. each domain in the dialogue is evaluated separately.

Prompt-recoverable errors can be likely fixed by specific prompt engineering with some effort. These kinds of errors happen with all of the tested models. Examples of such errors are invalid structure of the generated dialogue state, copying slot values instead of using canonical values from the ontology, failure to delexicalize some of the values, etc. Most of these errors can also be fixed in postprocessing – for example, we can employ more robust parsers or fuzzy matching of slot values.

Inherent errors, on the other hand, are likely not easily fixable by prompt modifications. They are not distributed evenly across the tested models and seem to constitute a more challenging problem.

Perhaps the most important error, common to all the models, is hallucination, i.e., the model's output responses are not grounded in the context (such as offering entities that are not included in the database results). This happens in about 10-20% of the inspected dialogues. Some models (*GPT-NeoXT, OPT-IML*) tend to generate more content than requested. This happens in more than 50% of their failed dialogues. This sometimes means continuing the conversation for a few more turns (including hallucinating user turns), but the models also often generate unrelated text or even code snippets, see Table 7.7. With *Tk-Instruct*, we observed that in ca. 10% cases, it copies the belief state from the example given in the prompt instead of generating a relevant one. Another issue is that the models tend to repeat their previous responses as illustrated in Table 7.9.

Goal	`You are looking for information in Cambridge`
	`You are looking for places to go in town.`
	`The attraction should be a swimming pool and`
	`should be in the north.`
	`Make sure you get entrance fee.`
	`You are also looking for a train.`
	`The train should go to Cambridge and leave on Saturday.`
	`The train should depart from London Liverpool Street`
	`and should arrive by 20:00.`
	`Make sure you get travel time and train ID.`

Customer	`I would like to go swimming in the northern cambridge`

Assistant	`There is a swimming pool in the north.`
	`Would you like more information on it?`

Customer	`I would like to know the entrance fee`

Assistant	`The entrance fee for the pool is 4 GBP.`
	`Can i help you with anything else?`

Customer	`I am looking for a train from`
	`london liverpool street to cambridge.`
	`i would like to get there by sunday 20:00`

Assistant	`you can take train which arrives at Cambridge at 19:07`
	`The price of the ticket is 3 GBP.`

Customer	`what is the train id and the travel time?`
Assistant	`TR2855 arrives at Cambridge at 19:07 on sunday.`
Customer	`and the travel time`
Assistant	`TR2855 arrives at Cambridge at 19:07 on sunday.`
Customer	`how long will it take?`
Assistant	`it will take you 88 minutes to get there.`

Table 7.9: A sample evaluation conversation between human and LLM-based (ChatGPT) agent. The sample includes a textual goal description that instructs the user. The slot values are lexicalized in the assistant's responses. Although the systems fails initially and the customer needs to ask repeatedly about the duration of travel, it can get the desired information in the end. The slot values are highlighted for both informed and requested values.

7.6 Discussion

We present an experimental evaluation of instruction-tuned LLMs applied to the established task of task-oriented dialogue modeling, with five LLMs evaluated on two datasets. We find that current LLMs have difficulties concerning belief state tracking. The LLM-based state trackers struggle even when provided with in-context few-shot examples. Some problems arise because the LLM might not provide correctly formatted outputs. These errors might be accounted for by restricting the decoding techniques and robust parsing mechanisms. On the other hand, a non-trivial portion of errors come from hallucinated slot values that are not present in the conversation. This behavior is harder to address successfully.

If provided with a correct belief state, the models can interact with the user well, provide useful information, and fulfill the user's needs. While the performance does not match the supervised state of the art on the evaluated datasets, it is important to note that these models were not fine-tuned on in-domain data and work with just a domain description or a few examples (which improves performance). Moreover, few-shot examples improve the model performance and access to the oracle belief state. Therefore, carefully picking representative examples and combining the LLM with an in-domain belief tracker can be a viable choice for a task-oriented dialogue pipeline. Also, our evaluation experiments suggest that LLM-based systems are surprisingly good in human interaction, surpassing the results suggested by the corpus-based automatic evaluation metrics.

Limitations One of the drawbacks of our work is the reproducibility. Closed models like ChatGPT are available only via API, and their behavior can change with time as out-of-date model are deprecated, making the experiments impossible to reproduce. To address this, we include more models for the evaluation. Another problem is the rapid development in this field of study. New and stronger LLMs frequently appear, quickly making some of our results obsolete. However, our experiments aim to put the LLM performance in context and show that despite their great capabilities, the approach of in-context learning is unlikely to solve the problem of task-oriented dialogues in the near future.

It would also be desirable to focus more on prompt engineering techniques since the LLMs are arguably sensitive to the choice of the right prompt, which is model-specific.

8 Conclusion

In this thesis, we discussed the problems of data annotation needed for task-oriented dialogue modeling. We proposed multiple approaches to address how current systems rely on extensive data annotation. Specifically, we explored a way to mitigate the need for data annotation in Chapter 4 and discuss this in Section 8.1. We also proposed novel approaches to dialogue modeling in Chapters 5, 6 and 7. Some results are very promising and competitive with state-of-the-art, but we also present some negative results. We conclude from these experiments in Sections 8.3 and 8.2. Finally, we discuss possible future research directions in this field in Section 8.4.

8.1 Decreasing the amount of required supervision

Automatic data labeling procedures are desirable since they can save expensive resources by providing valuable insights into the gathered conversation data. In Chapter 4, we proposed a pipeline method for the unsupervised discovery of dialogue slot schema and automatic labeling. The method iteratively refines a set of input candidates to obtain semantically coherent concepts that are suitable to use as slots to guide a dialogue system in a particular domain. The candidates are identified with arbitrary generic open-domain taggers such as NER or semantic parsers. The pipeline is generic because it can take an arbitrary set of input candidates, regardless of the candidate identification method. We showed that the method can successfully exploit the inputs and refine the initial candidate pool to outperform other approaches. Moreover, we have shown that the outputs of our pipeline can be used as a noisy supervision for training task-oriented dialogue system models.

8.2 Training dialogues from unlabeled data

The ultimate goal is to train task-oriented dialogue models solely from an unlabeled input corpus. As an intermediate step toward this goal, we introduced a novel usage of latent variable models for TOD generation in Chapter 5. In particular, we focused on latent system action modeling and providing the model with a means to communicate with external interfaces using sparse annotation. Modeling task-oriented dialogue this way is challenging, and achieving competitive performance with state-of-the-art supervised models is hard. However, our latent action models based on variational training show promising performance when outperforming other baseline approaches, even with many more parameters when the same amount of training data is presented. Moreover, our model creates representations in a discrete latent space that can be used to predict the system's actions successfully.

8.3 Less data for end-to-end TOD models

Another direction is using full supervision but minimizing the required training data. We explored this path in Chapters 6 and 7. First, we investigated to what extent pre-trained language models can transfer the obtained knowledge to previously unseen domains when initially fine-tuning them on different data. The power of pre-trained large language models is great, and they can achieve good performance with only a fraction of the training set available to them. We observed the importance of examples in the in-context learning approach. Although LLMs seem to struggle with the belief state tracking task, this can likely be fixed with fine-tuning techniques or different task formulation. Moreover, based on our observations in human evaluation, using LLMs promises to bridge the gap between academic datasets and real-world use cases as LLMs can handle various conversation behaviors very well.

8.4 Future research directions

Thanks to the tremendous recent progress in large language models pre-training and applications, unsupervised dialogue modeling will likely improve greatly in the near future. LLMs can be used to obtain better representations and, together with contrastive learning objectives, achieve previously unseen performance in dialogue structure discovery and schema induction. Therefore, our method proposed in Chapter 4 can be improved with stronger and more capable initial candidate identification and representation models. Moreover, our method works globally for arbitrarily large datasets. With the growing size of the context window that can be input to LLMs, we can explore approaches that analyze much larger subsets of datasets simultaneously, thus providing more context and information for the model decision.

The modeling capabilities of the Transformer-based models will likely achieve significant improvements in human-machine interaction with little to no need to provide in-domain data examples. Their performance can contribute to creating more powerful latent variable models, which we discussed in Chapter 5. First, submodules of the pre-trained models could be used to model more capable systems based on the VAE architecture, as proposed, for example, in Li et al. (2020a). Moreover, the low-parameter fine-tuning approaches such as LoRA (Hu et al., 2021) can be modified similarly to VAE and model distribution parameters. We can then sample from these predicted distributions and perhaps introduce an additional control over the model behavior and explain its actions.

As for using the pre-trained LLMs, which we discussed in Chapter 7, we can expect a lot of applications based on in-context learning augmented with retrieval mechanisms to guide the model with certain dialogue flows. For task-oriented dialogue, in particular, some challenges need to be addressed, such as the tendency of LLMs to produce answers not grounded in the context introduced in prompt, so-called hallucinations. The models also tend to deviate from the desired output structure, which can be addressed by finetuning them with proper data. Nevertheless, we might experience a paradigm shift when a possible future architecture communicates with APIs directly via in-line function calls instead of explicitly modeling the belief state, similar to the approach proposed in Schick et al. (2023). Apart from the performance gains, this would greatly simplify the interaction with various interfaces.

One way or the other, we are living in exciting times, not only for NLP but also for image processing, multimodal systems, embodied agents, speech technologies, and much more. As these technologies integrate more and more, there is an even more exciting future ahead of us.

Bibliography

Adel, H. – Roth, B. – Schütze, H. Comparing convolutional neural networks to traditional models for slot filling. *arXiv preprint arXiv:1603.05157*. 2016.

Bahdanau, D. – Cho, K. – Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*. 2014, abs/1409.0473. ISSN 2331-8422.

Baker, C. F. – Fillmore, C. J. – Lowe, J. B. The berkeley framenet project. In *Proceedings of COLING*, p. 86–90, 1998.

Balakrishnan, A. – Rao, J. – Upasani, K. – White, M. – Subba, R. Constrained Decoding for Neural NLG from Compositional Representations in Task-Oriented Dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, p. 831–844, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1080. Available at: `https://aclanthology.org/P19-1080`.

Bao, S. – He, H. – Wang, F. – Wu, H. PLATO: Pre-trained Dialogue Generation Model with Discrete Latent Variable. *arXiv preprint arXiv:1910.07931*. 2019.

Bapna, A. – Tür, G. – Hakkani-Tür, D. – Heck, L. Towards Zero-Shot Frame Semantic Parsing for Domain Scaling. In *Proceedings of Interspeech 2017*, p. 2476–2480, August 2017. doi: 10.21437/Interspeech.2017-518. Available at: `http://www.isca-speech.org/archive/Interspeech_2017/abstracts/0518.html`.

Berners-Lee, T. – Hendler, J. – Lassila, O. – others. The semantic web. *Scientific American*. 2001, 284, 5, p. 28–37.

Black, S. – Biderman, S. – Hallahan, E. – Anthony, Q. – Gao, L. – Golding, L. – He, H. – Leahy, C. – McDonell, K. – Phang, J. – others. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*. 2022.

Bojanowski, P. – Grave, E. – Joulin, A. – Mikolov, T. Enriching word vectors with subword information. *Transactions of the ACL*. 2017, 5, p. 135–146. Available at: `https://www.aclweb.org/anthology/Q17-1010/`.

Bordes, A. – Boureau, Y. – Weston, J. Learning End-to-End Goal-Oriented Dialog. In *5th International Conference on Learning Representations, ICLR 2017*, Toulon, France, 2017. Available at: `https://openreview.net/forum?id=S1Bb3D5gg`.

BOWMAN, S. R. – VILNIS, L. – VINYALS, O. – DAI, A. M. – JOZEFOWICZ, R. – BENGIO, S. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*. 2015.

BOWMAN, S. R. – VILNIS, L. – VINYALS, O. – DAI, A. – JOZEFOWICZ, R. – BENGIO, S. Generating Sentences from a Continuous Space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, p. 10–21, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1002. Available at: https://aclanthology.org/K16-1002.

BROWN, T. – MANN, B. – RYDER, N. – SUBBIAH, M. – KAPLAN, J. D. – DHARIWAL, P. – NEELAKANTAN, A. – SHYAM, P. – SASTRY, G. – ASKELL, A. – OTHERS. Language models are few-shot learners. *Advances in neural information processing systems*. 2020, 33, p. 1877–1901.

BRYCHCÍN, T. – KRÁL, P. Unsupervised dialogue act induction using gaussian mixtures. *arXiv preprint arXiv:1612.06572*. 2016.

BUDZIANOWSKI, P. – WEN, T.-H. – TSENG, B.-H. – CASANUEVA, I. – ULTES, S. – RAMADAN, O. – GAŠIĆ, M. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 5016–5026, Brussels, Belgium, October-November 2018a. Association for Computational Linguistics. doi: 10.18653/v1/D18-1547. Available at: https://aclanthology.org/D18-1547.

BUDZIANOWSKI, P. – WEN, T.-H. – TSENG, B.-H. – CASANUEVA, I. – ULTES, S. – RAMADAN, O. – GAŠIĆ, M. MultiWOZ – a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of EMNLP*, 2018b. Available at: https://www.aclweb.org/anthology/D18-1547.

CAI, D. – WANG, Y. – BI, W. – TU, Z. – LIU, X. – SHI, S. Retrieval-guided Dialogue Response Generation via a Matching-to-Generation Framework. 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)*, p. 1866–1875, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1195. Available at: https://aclanthology.org/D19-1195.

CALLISON-BURCH, C. – OSBORNE, M. – KOEHN, P. Re-evaluating the Role of Bleu in Machine Translation Research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, p. 249–256, Trento, Italy, April 2006. Association for Computational Linguistics. Available at: https://aclanthology.org/E06-1032.

Chao, G.-L. – Lane, I. BERT-DST: Scalable End-to-End Dialogue State Tracking with Bidirectional Encoder Representations from Transformer. *arXiv preprint arXiv:1910.07931*. 2019.

Chen, M. – Tworek, J. – Jun, H. – Yuan, Q. – Pinto, H. P. d. O. – Kaplan, J. – Edwards, H. – Burda, Y. – Joseph, N. – Brockman, G. – others. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*. 2021.

Chen, Y.-N. – Wang, W. Y. – Rudnicky, A. I. Leveraging frame semantics and distributional semantics for unsupervised semantic slot induction in spoken dialogue systems. In *Proceedings of IEEE SLT*, p. 584–589, 2014. doi: 10.1109/SLT.2014.7078639.

Chen, Y.-N. – Wang, W. Y. – Rudnicky, A. Jointly modeling inter-slot relations by random walk on knowledge graphs for unsupervised spoken language understanding. In *Proc. NAACL*, p. 619–629, 2015.

Chen, Y.-N. – Hakkani-Tür, D. – He, X. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *Proc. IEEE ICASSP*, p. 6045–6049, 2016.

Cho, K. – Merriënboer, B. – Bahdanau, D. – Bengio, Y. 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*, p. 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012. Available at: `https://aclanthology.org/W14-4012`.

Christiano, P. F. – Leike, J. – Brown, T. – Martic, M. – Legg, S. – Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*. 2017, 30.

Chung, J. – Kastner, K. – Dinh, L. – Goel, K. – Courville, A. C. – Bengio, Y. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, p. 2980–2988, 2015.

Coope, S. – Farghly, T. – Gerz, D. – Vulić, I. – Henderson, M. Span-ConveRT: Few-shot Span Extraction for Dialog with Pretrained Conversational Representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 107–121, Online, July 2020. Available at: `https://www.aclweb.org/anthology/2020.acl-main.11`.

Core, M. G. – Allen, J. Coding dialogs with the DAMSL annotation scheme. In ., 1997.

Das, D. – Schneider, N. – Chen, D. – Smith, N. A. Probabilistic Frame-semantic Parsing. In *Proceedings of NAACL-HLT*, p. 948–956, Los Angeles, California, 2010. Available at: https://www.aclweb.org/anthology/N10-1138.

Deriu, J. – Rodrigo, A. – Otegi, A. – Echegoyen, G. – Rosset, S. – Agirre, E. – Cieliebak, M. Survey on Evaluation Methods for Dialogue Systems. *Artificial Intelligence Review*. January 2021, 54, p. 755–810. doi: 10.1007/s10462-020-09866-x. Available at: http://arxiv.org/abs/1905.04071.

Devlin, J. – Chang, M.-W. – Lee, K. – Toutanova, K. 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)*, p. 4171–4186, Minneapolis, Minnesota, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. Available at: https://aclanthology.org/N19-1423.

Devlin, J. – Chang, M.-W. – Lee, K. – Toutanova, K. 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 (NAACL-HLT)*, p. 4171–4186, Minneapolis, MN, USA, 2019b. doi: 10.18653/v1/N19-1423. Available at: https://www.aclweb.org/anthology/N19-1423.

Dong, Q. – Li, L. – Dai, D. – Zheng, C. – Wu, Z. – Chang, B. – Sun, X. – Xu, J. – Sui, Z. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*. 2022.

Eric, M. – Krishnan, L. – Charette, F. – Manning, C. D. Key-Value Retrieval Networks for Task-Oriented Dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, p. 37–49, Saarbrücken, Germany, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5506. Available at: https://aclanthology.org/W17-5506.

Eric, M. – Goel, R. – Paul, S. – Sethi, A. – Agarwal, S. – Gao, S. – Kumar, A. – Goyal, A. – Ku, P. – Hakkani-Tur, D. MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines. In *Proceedings of the 12th Language Resources and Evaluation Conference*, p. 422–428, Marseille, France, May 2020. Available at: https://www.aclweb.org/anthology/2020.lrec-1.53.

Feng, Y. – Yang, S. – Zhang, S. – Zhang, J. – Xiong, C. – Zhou, M. – Wang, H. Fantastic Rewards and How to Tame Them: A Case Study on Reward Learning for Task-oriented Dialogue Systems. *arXiv preprint arXiv:2302.10342*. 2023.

Feng, Y. – Wang, Y. – Li, H. A Sequence-to-Sequence Approach to Dialogue State Tracking. 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)*, p. 1714–1725, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.135. Available at: `https://aclanthology.org/2021.acl-long.135`.

Fu, H. – Li, C. – Liu, X. – Gao, J. – Celikyilmaz, A. – Carin, L. Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing. 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)*, p. 240–250, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1021. Available at: `https://aclanthology.org/N19-1021`.

Gardner, M. – Grus, J. – Neumann, M. – Tafjord, O. – Dasigi, P. – Liu, N. F. – Peters, M. – Schmitz, M. – Zettlemoyer, L. S. AllenNLP: A Deep Semantic Natural Language Processing Platform. In *Proceedings of ACL Workshop for NLP Open Source Software (NLP-OSS)*, 2017. Available at: `https://www.aclweb.org/anthology/W18-2501/`.

Gašić, M. – Young, S. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. 2013, 22, 1, p. 28–40.

Gašić, M. – Jurčíček, F. – Keizer, S. – Mairesse, F. – Thomson, B. – Yu, K. – Young, S. Gaussian processes for fast policy optimisation of POMDP-based dialogue managers. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, p. 201–204. Association for Computational Linguistics, 2010.

Golchin, S. – Surdeanu, M. Time Travel in LLMs: Tracing Data Contamination in Large Language Models. *arXiv preprint arXiv:2308.08493*. 2023.

Goo, C.-W. – Gao, G. – Hsu, Y.-K. – Huo, C.-L. – Chen, T.-C. – Hsu, K.-W. – Chen, Y.-N. Slot-Gated Modeling for Joint Slot Filling and Intent Prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, p. 753–757, New Orleans, Louisiana, June 2018. doi: 10.18653/v1/N18-2118. Available at: `https://www.aclweb.org/anthology/N18-2118`.

Goodfellow, I. J. – Bengio, Y. – Courville, A. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Goutte, C. – Gaussier, E. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In *European conference on information retrieval*, p. 345–359. Springer, 2005.

Gunasekara, C. – Kim, S. – D'Haro, L. F. – Rastogi, A. – Chen, Y.-N. – Eric, M. – Hedayatnia, B. – Gopalakrishnan, K. – Liu, Y. – Huang, C.-W. – others. Overview of the ninth dialog system technology challenge: Dstc9. *arXiv preprint arXiv:2011.06486*. 2020.

Heck, L. – Hakkani-Tür, D. Exploiting the semantic web for unsupervised spoken language understanding. In *Proceedings of IEEE SLT*, p. 228–233, 2012. doi: 10.1109/SLT.2012.6424227.

Hemphill, C. T. – Godfrey, J. J. – Doddington, G. R. The ATIS spoken language systems pilot corpus. In *Proceedings of the workshop on Speech and Natural Language - HLT '90*, p. 96–101, Hidden Valley, Pennsylvania, 1990. doi: 10.3115/116580.116613. Available at: `https://www.aclweb.org/anthology/H90-1021/`.

Henderson, M. – Gašić, M. – Thomson, B. – Tsiakoulis, P. – Yu, K. – Young, S. Discriminative spoken language understanding using word confusion networks. In *Proceedings of IEEE SLT*, p. 176–181, 2012. doi: 10.1109/SLT.2012.6424218.

Henderson, M. – Thomson, B. – Young, S. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, p. 360–365, December 2014. doi: 10.1109/SLT.2014.7078601.

Hochreiter, S. – Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* November 1997, 9, 8, p. 1735–1780. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. Available at: `https://doi.org/10.1162/neco.1997.9.8.1735`.

Hoffmann, J. – Borgeaud, S. – Mensch, A. – Buchatskaya, E. – Cai, T. – Rutherford, E. – Casas, D. d. L. – Hendricks, L. A. – Welbl, J. – Clark, A. – others. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*. 2022.

Howard, J. – Ruder, S. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. Available at: `https://aclanthology.org/P18-1031`.

Hu, E. J. – Shen, Y. – Wallis, P. – Allen-Zhu, Z. – Li, Y. – Wang, S. – Wang, L. – Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*. 2021.

Hu, Y. – Lee, C.-H. – Xie, T. – Yu, T. – Smith, N. A. – Ostendorf, M. In-Context Learning for Few-Shot Dialogue State Tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, p. 2627–2643, Abu Dhabi, United Arab Emirates, December 2022a. Association for Computational Linguistics. Available at: `https://aclanthology.org/2022.findings-emnlp.193`.

Hu, Y. – Lee, C.-H. – Xie, T. – Yu, T. – Smith, N. A. – Ostendorf, M. In-context learning for few-shot dialogue state tracking. *arXiv preprint arXiv:2203.08568*. 2022b.

Huang, T. – Halbe, S. A. – Sankar, C. – Amini, P. – Kottur, S. – Geramifard, A. – Razaviyayn, M. – Beirami, A. Robustness through Data Augmentation Loss Consistency. *Transactions on Machine Learning Research*. 2023. Available at: `https://openreview.net/forum?id=a1meaRy1bN`.

Huang, X. – Qi, J. – Sun, Y. – Zhang, R. MALA: Cross-Domain Dialogue Generation with Action Learning. *arXiv preprint arXiv:1912.08442*. 2019.

Huang, X. – Qi, J. – Sun, Y. – Zhang, R. MALA: Cross-Domain Dialogue Generation with Action Learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, p. 7977–7984. AAAI Press, 2020. Available at: `https://ojs.aaai.org/index.php/AAAI/article/view/6306`.

Hudeček, V. – Dušek, O. Learning Interpretable Latent Dialogue Actions With Less Supervision. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, p. 297–308, Online only, November 2022. Association for Computational Linguistics. Available at: `https://aclanthology.org/2022.aacl-main.24`.

Hudeček, V. – Dušek, O. Are Large Language Models All You Need for Task-Oriented Dialogue? In *Proceedings of the 24th Meeting of the Special Interest Group on Discourse and Dialogue*, p. 216–228, Prague, Czechia, September 2023. Association for Computational Linguistics. Available at: `https://aclanthology.org/2023.sigdial-1.21`.

Hudeček, V. – Dušek, O. – Yu, Z. Discovering Dialogue Slots with Weak Supervision. 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)*, p. 2430–2442, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.189. Available at: `https://aclanthology.org/2021.acl-long.189`.

Hudeček, V. – Schaub, L.-p. – Stancl, D. – Paroubek, P. – Dušek, O. A Unifying View On Task-oriented Dialogue Annotation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, p. 1286–1296, Marseille, France, June 2022. European Language Resources Association. Available at: `https://aclanthology.org/2022.lrec-1.137`.

Iizuka, S. – Mochizuki, S. – Ohashi, A. – Yamashita, S. – Guo, A. – Higashinaka, R. Clarifying the Dialogue-Level Performance of GPT-3.5 and GPT-4 in Task-Oriented and Non-Task-Oriented Dialogue Systems. 2023.

Iyer, S. – Lin, X. V. – Pasunuru, R. – Mihaylov, T. – Simig, D. – Yu, P. – Shuster, K. – Wang, T. – Liu, Q. – Koura, P. S. – others. OPT-IML: Scaling Language Model Instruction Meta Learning through the Lens of Generalization. *arXiv preprint arXiv:2212.12017*. 2022.

Jang, E. – Gu, S. – Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*. 2016.

Jang, E. – Gu, S. – Poole, B. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations (ICLR 2017)*, 2017. Available at: `https://arxiv.org/abs/1611.01144`.

Jin, X. – Lei, W. – Ren, Z. – Chen, H. – Liang, S. – Zhao, Y. – Yin, D. Explicit State Tracking with Semi-Supervision for Neural Dialogue Generation. In *Proceedings of ACM CIKM*, p. 1403–1412, 2018. doi: 10.1145/3269206.3271683.

Johnson, J. – Douze, M. – Jégou, H. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*. 2019, 7, 3, p. 535–547.

Jurafsky, D. *Speech & language processing*. Pearson Education India, 2000.

Kale, M. – Rastogi, A. Template Guided Text Generation for Task-Oriented Dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 6505–6520, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.527. Available at: `https://aclanthology.org/2020.emnlp-main.527`.

Kaplan, J. – McCandlish, S. – Henighan, T. – Brown, T. B. – Chess, B. – Child, R. – Gray, S. – Radford, A. – Wu, J. – Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*. 2020.

Kelley, H. J. Gradient theory of optimal flight paths. *Ars Journal*. 1960, 30, 10, p. 947–954.

Kingma, D. P. – Welling, M. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*, Banff, AB, Canada, April 2014. Available at: `http://arxiv.org/abs/1312.6114`.

Kulhánek, J. – Hudeček, V. – Nekvinda, T. – Dušek, O. AuGPT: Auxiliary Tasks and Data Augmentation for End-To-End Dialogue with Pre-Trained Language Models. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, p. 198–210, Online, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.nlp4convai-1.19. Available at: `https://aclanthology.org/2021.nlp4convai-1.19`.

Lample, G. – Ballesteros, M. – Subramanian, S. – Kawakami, K. – Dyer, C. Neural architectures for named entity recognition. In *Proc. NAACL*, 2016.

Lee, C.-H. – Cheng, H. – Ostendorf, M. Dialogue State Tracking with a Language Model using Schema-Driven Prompting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, p. 4937–4949, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.404. Available at: `https://aclanthology.org/2021.emnlp-main.404`.

Lei, W. – Jin, X. – Kan, M.-Y. – Ren, Z. – He, X. – Yin, D. Sequicity: Simplifying Task-oriented Dialogue Systems with Single Sequence-to-Sequence Architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 1437–1447, Melbourne, Australia, 2018a. doi: 10.18653/v1/P18-1133. Available at: `https://www.aclweb.org/anthology/P18-1133`.

Lei, W. – Jin, X. – Kan, M.-Y. – Ren, Z. – He, X. – Yin, D. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 1437–1447, 2018b.

Levesque, H. – Davis, E. – Morgenstern, L. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

Lewis, M. – Liu, Y. – Goyal, N. – Ghazvininejad, M. – Mohamed, A. – Levy, O. – Stoyanov, V. – Zettlemoyer, L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. Available at: `https://aclanthology.org/2020.acl-main.703`.

Li, B. – He, J. – Neubig, G. – Berg-Kirkpatrick, T. – Yang, Y. A surprisingly effective fix for deep latent variable modeling of text. *arXiv preprint arXiv:1909.00868*. 2019.

Li, C. – Gao, X. – Li, Y. – Peng, B. – Li, X. – Zhang, Y. – Gao, J. Optimus: Organizing Sentences via Pre-trained Modeling of a Latent Space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 4678–4699, Online, November 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.378. Available at: `https://aclanthology.org/2020.emnlp-main.378`.

Li, X. – Chen, Y.-N. – Li, L. – Gao, J. – Celikyilmaz, A. End-to-End Task-Completion Neural Dialogue Systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, p. 733–743, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. Available at: `https://aclanthology.org/I17-1074`.

Li, Z. – Murkute, J. V. – Gyawali, P. K. – Wang, L. Progressive learning and disentanglement of hierarchical representations. *arXiv preprint arXiv:2002.10549*. 2020b.

Liang, W. – Tian, Y. – Chen, C. – Yu, Z. MOSS: End-to-End Dialog System Framework with Modular Supervision. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, p. 8327–8335. AAAI Press, 2020. Available at: `https://ojs.aaai.org/index.php/AAAI/article/view/6349`.

Lin, Z. – Madotto, A. – Winata, G. I. – Fung, P. MinTL: Minimalist Transfer Learning for Task-Oriented Dialogue Systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 3391–3405, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.273. Available at: `https://aclanthology.org/2020.emnlp-main.273`.

Liu, B. – Lane, I. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*. 2016.

Liu, Y. – Ott, M. – Goyal, N. – Du, J. – Joshi, M. – Chen, D. – Levy, O. – Lewis, M. – Zettlemoyer, L. – Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*. 2019.

Liu, Z. – Winata, G. I. – Xu, P. – Fung, P. Coach: A Coarse-to-Fine Approach for Cross-domain Slot Filling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, p. 19–25, Online, July 2020. Available at: https://www.aclweb.org/anthology/2020.acl-main.3.

Loshchilov, I. – Hutter, F. Decoupled Weight Decay Regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. Available at: https://openreview.net/forum?id=Bkg6RiCqY7.

Lowe, R. – Noseworthy, M. – Serban, I. V. – Angelard-Gontier, N. – Bengio, Y. – Pineau, J. 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)*, p. 1116–1126, 2017.

Lu, B.-R. – Hu, Y. – Cheng, H. – Smith, N. A. – Ostendorf, M. Unsupervised Learning of Hierarchical Conversation Structure. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, p. 5657–5670, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. Available at: https://aclanthology.org/2022.findings-emnlp.415.

Lubis, N. – Geishauser, C. – Heck, M. – Lin, H.-c. – Moresi, M. – Niekerk, C. – Gasic, M. LAVA: Latent Action Spaces via Variational Auto-encoding for Dialogue Policy Optimization. In *Proceedings of the 28th International Conference on Computational Linguistics*, p. 465–479, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.41. Available at: https://aclanthology.org/2020.coling-main.41.

Lubis, N. – Geishauser, C. – Lin, H.-c. – Niekerk, C. – Heck, M. – Feng, S. – Gasic, M. Dialogue Evaluation with Offline Reinforcement Learning. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, p. 478–489, Edinburgh, UK, September 2022. Association for Computational Linguistics. Available at: https://aclanthology.org/2022.sigdial-1.46.

Luong, T. – Pham, H. – Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, p. 1412–1421, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1166. Available at: https://aclanthology.org/D15-1166.

Madotto, A. – Wu, C.-S. – Fung, P. Mem2Seq: Effectively Incorporating Knowledge Bases into End-to-End Task-Oriented Dialog Systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 1468–1478, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1136. Available at: `https://aclanthology.org/P18-1136`.

Madotto, A. – Liu, Z. – Lin, Z. – Fung, P. Language models as few-shot learner for task-oriented dialogue systems. *arXiv preprint arXiv:2008.06239*. 2020.

Mairesse, F. – Young, S. Stochastic Language Generation in Dialogue using Factored Language Models. *Computational Linguistics*. December 2014, 40, 4, p. 763–799. doi: 10.1162/COLI_a_00199. Available at: `https://aclanthology.org/J14-4003`.

Manning, C. D. – Schütze, H. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999. Available at: `http://nlp.stanford.edu/fsnlp/`.

McTear, M. *Conversational ai: Dialogue systems, conversational agents, and chatbots*. Springer Nature, 2022.

Mesnil, G. – Dauphin, Y. – Yao, K. – Bengio, Y. – Deng, L. – Hakkani-Tur, D. – He, X. – Heck, L. – Tur, G. – Yu, D. – others. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. 2014, 23, 3, p. 530–539.

Micikevicius, P. – Narang, S. – Alben, J. – Diamos, G. – Elsen, E. – Garcia, D. – Ginsburg, B. – Houston, M. – Kuchaiev, O. – Venkatesh, G. – others. Mixed precision training. *arXiv preprint arXiv:1710.03740*. 2017.

Mihalcea, R. – Tarau, P. Textrank: Bringing order into text. In *Proc. EMNLP*, 2004.

Mikolov, T. – Karafiát, M. – Burget, L. – Černockỳ, J. – Khudanpur, S. Recurrent neural network based language model. *Interspeech 2010*. 2010.

Mikolov, T. – Sutskever, I. – Chen, K. – Corrado, G. S. – Dean, J. Distributed representations of words and phrases and their compositionality. In *Proceedings of NeurIPS*, p. 3111–3119, 2013.

Min, S. – Lyu, X. – Holtzman, A. – Artetxe, M. – Lewis, M. – Hajishirzi, H. – Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*. 2022.

Mosbach, M. – Pimentel, T. – Ravfogel, S. – Klakow, D. – Elazar, Y. Few-shot Fine-tuning vs. In-context Learning: A Fair Comparison and Evaluation. In *Findings of the Association for Computational Linguistics: ACL 2023*, p. 12284–12314, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.779. Available at: `https://aclanthology.org/2023.findings-acl.779`.

Mrkšić, N. – Séaghdha, D. O. – Wen, T.-H. – Thomson, B. – Young, S. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of ACL*, p. 1777–1788, 2017. Available at: `https://www.aclweb.org/anthology/P17-1163`.

Mukherjee, S. – Hudeček, V. – Dušek, O. Polite Chatbot: A Text Style Transfer Application. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, p. 87–93, 2023.

Neil, D. – Pfeiffer, M. – Liu, S.-C. Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences. In Lee, D. D. – Luxburg, U. – Garnett, R. – Sugiyama, M. – Guyon, I. (Ed.) *Advances in Neural Information Processing Systems 29*, p. 3889 – 3898, Red Hook, NY, 2017. Curran. ISBN 978-1-5108-3881-9, 30th Annual Conference on Neural Information Processing Systems (NIPS 2016); Conference Location: Barcelona, Spain; Conference Date: December 5-10, 2016; Poster presentation.

Nekvinda, T. – Dušek, O. Shades of BLEU, Flavours of Success: The Case of MultiWOZ. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, p. 34–46, Online, August 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.gem-1.4. Available at: `https://aclanthology.org/2021.gem-1.4`.

Nekvinda, T. – Dušek, O. AARGH! End-to-end Retrieval-Generation for Task-Oriented Dialog. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, p. 283–297, Edinburgh, UK, September 2022. Association for Computational Linguistics. Available at: `https://aclanthology.org/2022.sigdial-1.29`.

Nekvinda, T. – Dušek, O. Shades of BLEU, Flavours of Success: The Case of MultiWOZ. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, p. 34–46, Online, August 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.gem-1.4. Available at: `https://aclanthology.org/2021.gem-1.4`.

Oh, A. – Rudnicky, A. Stochastic language generation for spoken dialogue systems. In *ANLP-NAACL 2000 Workshop: Conversational Systems*, 2000.

Ouyang, L. – Wu, J. – Jiang, X. – Almeida, D. – Wainwright, C. – Mishkin, P. – Zhang, C. – Agarwal, S. – Slama, K. – Ray, A. – others. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*. 2022, 35, p. 27730–27744.

Page, L. – Brin, S. – Motwani, R. – Winograd, T. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Pandey, G. – Contractor, D. – Kumar, V. – Joshi, S. Exemplar Encoder-Decoder for Neural Conversation Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 1329–1338, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1123. Available at: `https://aclanthology.org/P18-1123`.

Papineni, K. – Roukos, S. – Ward, T. – Zhu, W.-J. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, p. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. Available at: `https://aclanthology.org/P02-1040`.

Paszke, A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*. Vancouver, Canada: Curran Associates, Inc., 2019. p. 8024–8035. Available at: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

Peng, B. – Li, X. – Li, L. – Gao, J. – Celikyilmaz, A. – Lee, S. – Wong, K.-F. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. *arXiv preprint arXiv:1704.03084*. 2017.

Peng, B. – Li, C. – Li, J. – Shayandeh, S. – Liden, L. – Gao, J. Soloist: Building Task Bots at Scale with Transfer Learning and Machine Teaching. *Transactions of the Association for Computational Linguistics*. 2021a, 9, p. 807–824. doi: 10.1162/tacl_a_00399. Available at: `https://aclanthology.org/2021.tacl-1.49`.

Peng, B. – Li, C. – Li, J. – Shayandeh, S. – Liden, L. – Gao, J. SOLOIST: Building Task Bots at Scale with Transfer Learning and Machine Teaching. *Trans. Assoc. Comput. Linguistics*. 2021b, 9, p. 907–824. doi: 10.1162/tacl_a_00399. Available at: `https://doi.org/10.1162/tacl_a_00399`.

Peters, M. E. – Neumann, M. – Iyyer, M. – Gardner, M. – Clark, C. – Lee, K. – Zettlemoyer, L. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, p. 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. Available at: `https://aclanthology.org/N18-1202`.

Pfeiffer, J. – Rücklé, A. – Poth, C. – Kamath, A. – Vulić, I. – Ruder, S. – Cho, K. – Gurevych, I. AdapterHub: A Framework for Adapting Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, p. 46–54, 2020.

Pinar Saygin, A. – Cicekli, I. – Akman, V. Turing test: 50 years later. *Minds and machines*. 2000, 10, 4, p. 463–518.

Plátek, O. – Bělohlávek, P. – Hudeček, V. – Jurčíček, F. Recurrent neural networks for dialogue state tracking. *arXiv preprint arXiv:1606.08733*. 2016.

Plátek, O. – Hudeček, V. – Schmidtová, P. – Lango, M. – Dušek, O. Three Ways of Using Large Language Models to Evaluate Chat. *arXiv preprint arXiv:2308.06502*. 2023.

Qin, L. – Xu, X. – Che, W. – Zhang, Y. – Liu, T. Dynamic Fusion Network for Multi-Domain End-to-end Task-Oriented Dialog. In Jurafsky, D. – Chai, J. – Schluter, N. – Tetreault, J. R. (Ed.) *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020*, p. 6344–6354, Online, 2020. doi: 10.18653/v1/2020.acl-main.565. Available at: `https://doi.org/10.18653/v1/2020.acl-main.565`.

Qiu, L. – Zhao, Y. – Shi, W. – Liang, Y. – Shi, F. – Yuan, T. – Yu, Z. – Zhu, S.-C. Structured Attention for Unsupervised Dialogue Structure Induction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, p. 1889–1899, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.148. Available at: `https://aclanthology.org/2020.emnlp-main.148`.

Qiu, L. – Wu, C.-S. – Liu, W. – Xiong, C. Structure extraction in task-oriented dialogues with slot clustering. *arXiv preprint arXiv:2203.00073*. 2022.

Radford, A. – Narasimhan, K. – Salimans, T. – Sutskever, I. – others. Improving language understanding by generative pre-training. 2018.

Radford, A. – Wu, J. – Child, R. – Luan, D. – Amodei, D. – Sutskever, I. – others. Language models are unsupervised multitask learners. *OpenAI blog*. 2019, 1, 8, p. 9.

RAFFEL, C. – SHAZEER, N. – ROBERTS, A. – LEE, K. – NARANG, S. – MATENA, M. – ZHOU, Y. – LI, W. – LIU, P. J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*. 2020a, 21, 140, p. 1–67. Available at: http://jmlr.org/papers/v21/20-074.html.

RAFFEL, C. – SHAZEER, N. – ROBERTS, A. – LEE, K. – NARANG, S. – MATENA, M. – ZHOU, Y. – LI, W. – LIU, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*. 2020b, 21, 1, p. 5485–5551.

RAMSHAW, L. – MARCUS, M. Text Chunking using Transformation-Based Learning. In *Third Workshop on Very Large Corpora*, 1995. Available at: https://aclanthology.org/W95-0107.

RASTOGI, A. – HAKKANI-TÜR, D. – HECK, L. Scalable multi-domain dialogue state tracking. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, p. 561–568. IEEE, 2017.

REIMERS, N. – GUREVYCH, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019a. Available at: https://arxiv.org/abs/1908.10084.

REIMERS, N. – GUREVYCH, I. Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2020. Available at: https://arxiv.org/abs/2004.09813.

REIMERS, N. – GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*. 2019b.

ROSENBERG, A. – HIRSCHBERG, J. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, p. 410–420, Prague, Czech Republic, June 2007. Association for Computational Linguistics. Available at: https://aclanthology.org/D07-1043.

ROTHMAN, D. *Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more.* Packt Publishing Ltd, 2021.

Rudnicky, A. I. – Thayer, E. H. – Constantinides, P. C. – Tchou, C. – Shern, R. – Lenzo, K. A. – Xu, W. – Oh, A. Creating natural dialogs in the Carnegie Mellon Communicator system. In *Proceedings of the 6th European Conference on Speech Communication and Technology*, p. 1531–1534, 1999. Available at: `http://www.speech.cs.cmu.edu/Communicator/papers/Natural%20Dialogs2.pdf`.

Schapire, R. E. – Singer, Y. BoosTexter: A boosting-based system for text categorization. *Machine learning*. 2000, 39, 2-3, p. 135–168.

Schaub, L.-P. – Hudecek, V. – Stancl, D. – Dusek, O. – Paroubek, P. Définition et détection des incohérences du système dans les dialogues orientés tâche. (We present experiments on automatically detecting inconsistent behavior of task-oriented dialogue systems from the context). In *Actes de la 28e Conférence sur le Traitement Automatique des Langues Naturelles. Volume 1 : conférence principale*, p. 142–152, Lille, France, 6 2021. ATALA. Available at: `https://aclanthology.org/2021.jeptalnrecital-taln.13`.

Schick, T. – Dwivedi-Yu, J. – Dessì, R. – Raileanu, R. – Lomeli, M. – Zettlemoyer, L. – Cancedda, N. – Scialom, T. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*. 2023.

Serban, I. V. – Sordoni, A. – Bengio, Y. – Courville, A. – Pineau, J. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Shah, D. – Gupta, R. – Fayazi, A. – Hakkani-Tur, D. Robust Zero-Shot Cross-Domain Slot Filling with Example Values. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, p. 5484–5490, Florence, Italy, 2019. doi: 10.18653/v1/P19-1547. Available at: `https://www.aclweb.org/anthology/P19-1547`.

Shalyminov, I. – Sordoni, A. – Atkinson, A. – Schulz, H. Fast Domain Adaptation for Goal-Oriented Dialogue Using a Hybrid Generative-Retrieval Transformer. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 8039–8043, May 2020. doi: 10.1109/ICASSP40776.2020.9053599. ISSN: 2379-190X.

Shalyminov, I. – Lee, S. – Eshghi, A. – Lemon, O. Data-Efficient Goal-Oriented Conversation with Dialogue Knowledge Transfer Networks. 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)*, p. 1741–1751, Hong Kong, China, November 2019a. Association for Computational Linguistics. doi: 10.18653/v1/D19-1183. Available at: `https://aclanthology.org/D19-1183`.

Shalyminov, I. – Lee, S. – Eshghi, A. – Lemon, O. Few-Shot Dialogue Generation Without Annotated Data: A Transfer Learning Approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, p. 32–39, Stockholm, Sweden, September 2019b. Association for Computational Linguistics. doi: 10.18653/v1/W19-5904. Available at: `https://aclanthology.org/W19-5904`.

Shi, C. – Chen, Q. – Sha, L. – Li, S. – Sun, X. – Wang, H. – Zhang, L. Auto-Dialabel: Labeling Dialogue Data with Unsupervised Learning. In *Proc. EMNLP*, p. 684–689, 2018.

Shi, W. – Zhao, T. – Yu, Z. Unsupervised Dialog Structure Learning. *arXiv preprint arXiv:1904.03736*. 2019.

Shi, Y. – Yao, K. – Chen, H. – Yu, D. – Pan, Y.-C. – Hwang, M.-Y. Recurrent support vector machines for slot tagging in spoken language understanding. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 393–399, 2016.

Straka, M. – Mediankin, N. – Kocmi, T. – Žabokrtský, Z. – Hudeček, V. – Hajič, J. SumeCzech: Large Czech News-Based Summarization Dataset. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). Available at: `https://aclanthology.org/L18-1551`.

Su, H. – Kasai, J. – Wu, C. H. – Shi, W. – Wang, T. – Xin, J. – Zhang, R. – Ostendorf, M. – Zettlemoyer, L. – Smith, N. A. – others. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975*. 2022.

Su, P.-H. – Gasic, M. – Mrksic, N. – Rojas-Barahona, L. – Ultes, S. – Vandyke, D. – Wen, T.-H. – Young, S. On-line active reward learning for policy optimisation in spoken dialogue systems. *arXiv preprint arXiv:1605.07669*. 2016.

Sukhbaatar, S. – Weston, J. – Fergus, R. – others. End-to-end memory networks. *Advances in neural information processing systems*. 2015, 28.

Sun, H. – Bao, J. – Wu, Y. – He, X. Mars: Semantic-aware Contrastive Learning for End-to-End Task-Oriented Dialog. *arXiv preprint arXiv:2210.08917*. 2022.

Sun, K. – Moon, S. – Crook, P. – Roller, S. – Silvert, B. – Liu, B. – Wang, Z. – Liu, H. – Cho, E. – Cardie, C. Adding Chit-Chat to Enhance Task-Oriented Dialogues. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 1570–1583, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.124. Available at: `https://aclanthology.org/2021.naacl-main.124`.

SURENDRAN, D. – LEVOW, G.-A. Dialog act tagging with support vector machines and hidden Markov models. In *Ninth International Conference on Spoken Language Processing*, 2006.

SWAYAMDIPTA, S. – THOMSON, S. – DYER, C. – SMITH, N. A. Frame-semantic parsing with softmax-margin segmental RNNs and a syntactic scaffold. *arXiv:1706.09528*. 2017. Available at: `https://arxiv.org/abs/1706.09528`.

TAORI, R. – GULRAJANI, I. – ZHANG, T. – DUBOIS, Y. – LI, X. – GUESTRIN, C. – LIANG, P. – HASHIMOTO, T. B. Stanford Alpaca: An Instruction-following LLaMA model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

THOMSON, B. – YOUNG, S. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*. 2010, 24, 4, p. 562–588.

TOUVRON, H. – LAVRIL, T. – IZACARD, G. – MARTINET, X. – LACHAUX, M.-A. – LACROIX, T. – ROZIÈRE, B. – GOYAL, N. – HAMBRO, E. – AZHAR, F. – RODRIGUEZ, A. – JOULIN, A. – GRAVE, E. – LAMPLE, G. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*. 2023.

TU, L. – XIONG, C. – ZHOU, Y. Prompt-Tuning Can Be Much Better Than Fine-Tuning on Cross-lingual Understanding With Multilingual Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, p. 5478–5485, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. Available at: `https://aclanthology.org/2022.findings-emnlp.401`.

VAHDAT, A. – KAUTZ, J. NVAE: A deep hierarchical variational autoencoder. *Advances in neural information processing systems*. 2020, 33, p. 19667–19679.

OORD, A. – VINYALS, O. – OTHERS. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, p. 6306–6315, 2017.

MAATEN, L. – HINTON, G. Visualizing data using t-SNE. *Journal of machine learning research*. 2008, 9, 11.

VASWANI, A. – SHAZEER, N. – PARMAR, N. – USZKOREIT, J. – JONES, L. – GOMEZ, A. N. – KAISER, Ł. – POLOSUKHIN, I. Attention is All You Need. In *Advances in Neural Information Processing Systems 30*, p. 6000–6010, Long Beach, CA, USA, December 2017. Curran Associates, Inc.

WANG, A. – SINGH, A. – MICHAEL, J. – HILL, F. – LEVY, O. – BOWMAN, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*. 2018.

Wang, Y. – Mishra, S. – Alipoormolabashi, P. – Kordi, Y. – Mirzaei, A. – Arunkumar, A. – Ashok, A. – Dhanasekaran, A. S. – Naik, A. – Stap, D. – others. Super-NaturalInstructions:Generalization via Declarative Instructions on 1600+ Tasks. In *EMNLP*, 2022.

Wei, J. – Tay, Y. – Bommasani, R. – Raffel, C. – Zoph, B. – Borgeaud, S. – Yogatama, D. – Bosma, M. – Zhou, D. – Metzler, D. – others. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*. 2022.

Weisser, M. DART – The dialogue annotation and research tool. *Corpus Linguistics and Linguistic Theory*. 2016, 12, 2, p. 355–388. doi: doi:10.1515/cllt-2014-0051. Available at: `https://doi.org/10.1515/cllt-2014-0051`.

Wen, T.-H. – Gašić, M. – Kim, D. – Mrkšić, N. – Su, P.-H. – Vandyke, D. – Young, S. Stochastic Language Generation in Dialogue using Recurrent Neural Networks with Convolutional Sentence Reranking. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, p. 275–284, Prague, Czech Republic, September 2015a. Association for Computational Linguistics. doi: 10.18653/v1/W15-4639. Available at: `https://aclanthology.org/W15-4639`.

Wen, T.-H. – Gašić, M. – Mrkšić, N. – Su, P.-H. – Vandyke, D. – Young, S. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, p. 1711–1721, Lisbon, Portugal, September 2015b. Association for Computational Linguistics. doi: 10.18653/v1/D15-1199. Available at: `https://aclanthology.org/D15-1199`.

Wen, T.-H. – Gašić, M. – Mrkšić, N. – Rojas-Barahona, L. M. – Su, P.-H. – Vandyke, D. – Young, S. Multi-domain Neural Network Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 120–129, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1015. Available at: `https://aclanthology.org/N16-1015`.

Wen, T.-H. – Miao, Y. – Blunsom, P. – Young, S. Latent intention dialogue models. In *Proceedings of ICML*, p. 3732–3741, 2017a.

Wen, T.-H. – Vandyke, D. – Mrkšić, N. – Gašić, M. – Rojas-Barahona, L. M. – Su, P.-H. – Ultes, S. – Young, S. A Network-based End-to-End Trainable Task-oriented Dialogue System. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, p. 438–449, Valencia, Spain, April 2017b. Association for Computational Linguistics. Available at: `https://aclanthology.org/E17-1042`.

Wen, T.-H. – Vandyke, D. – Mrkšić, N. – Gašić, M. – Rojas-Barahona, L. M. – Su, P.-H. – Ultes, S. – Young, S. A Network-based End-to-End Trainable Task-oriented Dialogue System. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, p. 438–449, Valencia, Spain, April 2017c. Association for Computational Linguistics. Available at: `https://www.aclweb.org/anthology/E17-1042`.

Wen, T.-H. – Vandyke, D. – Mrksić, N. – Gašić, M. – Rojas-Barahona, L. M. – Su, P.-H. – Ultes, S. – Young, S. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*, p. 438–449, 2017d. Available at: `https://www.aclweb.org/anthology/E17-1042`.

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

Weston, J. – Chopra, S. – Bordes, A. Memory Networks. In Bengio, Y. – LeCun, Y. (Ed.) *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. Available at: `http://arxiv.org/abs/1410.3916`.

Williams, J. – Raux, A. – Ramachandran, D. – Black, A. The dialog state tracking challenge. In *Proceedings of SIGDIAL*, p. 404–413, 2013. Available at: `https://www.aclweb.org/anthology/W13-4065/`.

Williams, J. D. – Asadi, K. – Zweig, G. Hybrid Code Networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 665–677, Vancouver, Canada, July 2017a. Association for Computational Linguistics. doi: 10.18653/v1/P17-1062. Available at: `https://aclanthology.org/P17-1062`.

Williams, J. D. – Asadi, K. – Zweig, G. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*. 2017b.

Wolf, T. – Sanh, V. – Chaumond, J. – Delangue, C. TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents. *CoRR*. 2019, abs/1901.08149. Available at: `http://arxiv.org/abs/1901.08149`.

Xie, T. – Yang, X. – Lin, A. S. – Wu, F. – Hashimoto, K. – Qu, J. – Kang, Y. M. – Yin, W. – Wang, H. – Yavuz, S. – others. Converse: A Tree-Based Modular Task-Oriented Dialogue System. *arXiv preprint arXiv:2203.12187*. 2022.

Xu, P. – Sarikaya, R. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *2013 ieee workshop on automatic speech recognition and understanding*, p. 78–83. IEEE, 2013.

Yaman, S. – Deng, L. – Yu, D. – Wang, Y.-Y. – Acero, A. An integrative and discriminative technique for spoken utterance classification. *IEEE Transactions on Audio, Speech, and Language Processing*. 2008, 16, 6, p. 1207–1214.

Yang, Y. – Li, Y. – Quan, X. Ubar: Towards fully end-to-end task-oriented dialog system with gpt-2. In *Proceedings of the AAAI Conference on Artificial Intelligence*, p. 14230–14238, 2021.

Young, S. – Gasic, M. – Thomson, B. – Williams, J. POMDP-Based Statistical Spoken Dialog Systems: A Review. *Proceedings of the IEEE*. 2013, 101, 5, p. 1160–1179. ISSN 0018-9219. doi: 10.1109/JPROC.2012.2225812.

Yu, D. – Wang, M. – Cao, Y. – Shafran, I. – Shafey, L. – Soltau, H. Unsupervised Slot Schema Induction for Task-oriented Dialog. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, p. 1174–1193, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.86. Available at: `https://aclanthology.org/2022.naacl-main.86`.

Zhang, S. – Roller, S. – Goyal, N. – Artetxe, M. – Chen, M. – Chen, S. – Dewan, C. – Diab, M. – Li, X. – Lin, X. V. – others. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*. 2022.

Zhang, X. – Peng, B. – Li, K. – Zhou, J. – Meng, H. SGP-TOD: Building Task Bots Effortlessly via Schema-Guided LLM Prompting. *arXiv preprint arXiv:2305.09067*. 2023.

Zhang, Y. – Sun, S. – Galley, M. – Chen, Y.-C. – Brockett, C. – Gao, X. – Gao, J. – Liu, J. – Dolan, B. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*. 2019.

Zhang, Y. – Zhong, V. – Chen, D. – Angeli, G. – Manning, C. D. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, p. 35–45, 2017.

Zhao, T. – Eskenazi, M. Zero-Shot Dialog Generation with Cross-Domain Latent Actions. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, p. 1–10, Melbourne, Australia, July 2018a. Association for Computational Linguistics. doi: 10.18653/v1/W18-5001. Available at: `https://aclanthology.org/W18-5001`.

Zhao, T. – Eskenazi, M. Zero-Shot Dialog Generation with Cross-Domain Latent Actions. In *Proceedings of SIGDIAL*, p. 1–10, 2018b. Available at: `http://aclweb.org/anthology/W18-5001`.

ZHAO, T. – LEE, K. – ESKENAZI, M. Unsupervised Discrete Sentence Representation Learning for Interpretable Neural Dialog Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, p. 1098–1107, Melbourne, Australia, July 2018. Available at: `http://aclweb.org/anthology/P18-1101`.

ZHONG, V. – XIONG, C. – SOCHER, R. Global-locally self-attentive dialogue state tracker. *arXiv preprint arXiv:1805.09655*. 2018.

ZHU, Q. – GEISHAUSER, C. – LIN, H. – NIEKERK, C. – PENG, B. – ZHANG, Z. – HECK, M. – LUBIS, N. – WAN, D. – ZHU, X. – GAO, J. – GAŠIĆ, M. – HUANG, M. ConvLab-3: A Flexible Dialogue System Toolkit Based on a Unified Data Format. *arXiv preprint arXiv:2211.17148*. 2022. Available at: `http://arxiv.org/abs/2211.17148`.

ZIEGLER, D. M. – STIENNON, N. – WU, J. – BROWN, T. B. – RADFORD, A. – AMODEI, D. – CHRISTIANO, P. – IRVING, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*. 2019.

ŽILKA, L. – MAREK, D. – KORVAS, M. – JURCICEK, F. Comparison of bayesian discriminative and generative models for dialogue state tracking. In *Proceedings of the SIGDIAL 2013 Conference*, p. 452–456, 2013.

List of Tables

List of Figures

List of Publications

We first present list of publications in which the author of this thesis is the main author, followed with more publications to which the author contributed. The number of citations was computed using Google Scholar.

Total number of citations: 112

Hudeček, V. – Dušek, O. – Yu, Z. Discovering Dialogue Slots with Weak Supervision. 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)*, p. 2430–2442, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.189. Available at: `https://aclanthology.org/2021.acl-long.189`

- This work presents pipeline method for unsupervised slot discovery.
- Citations: 20

Hudeček, V. – Dušek, O. Are Large Language Models All You Need for Task-Oriented Dialogue? In *Proceedings of the 24th Meeting of the Special Interest Group on Discourse and Dialogue*, p. 216–228, Prague, Czechia, September 2023. Association for Computational Linguistics. Available at: `https://aclanthology.org/2023.sigdial-1.21`

- In this work we use large language models and in-context learning to model task-oriented dialogue.
- Citations: 8

Hudeček, V. – Dušek, O. Learning Interpretable Latent Dialogue Actions With Less Supervision. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, p. 297–308, Online only, November 2022. Association for Computational Linguistics. Available at: `https://aclanthology.org/2022.aacl-main.24`

- In this publication we introduce our dialogue model with variational training and discrete latent space.
- Citations: 1

Hudeček, V. – Schaub, L.-p. – Stancl, D. – Paroubek, P. – Dušek, O. A Unifying View On Task-oriented Dialogue Annotation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, p. 1286–1296, Marseille, France, June 2022. European Language Resources Association. Available at: `https://aclanthology.org/2022.lrec-1.137`

- In this work we present novel dialogue corpus and related experiments.
- Citations: 0

Kulhánek, J. – Hudeček, V. – Nekvinda, T. – Dušek, O. AuGPT: Auxiliary Tasks and Data Augmentation for End-To-End Dialogue with Pre-Trained Language Models. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, p. 198–210, Online, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.nlp4convai-1.19. Available at: `https://aclanthology.org/2021.nlp4convai-1.19`

- This work presents the AuGPT model and related experiments. It also describes the DSTC 8 challenge participation.
- Citations: 44

Straka, M. – Mediankin, N. – Kocmi, T. – Žabokrtský, Z. – Hudeček, V. – Hajič, J. SumeCzech: Large Czech News-Based Summarization Dataset. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). Available at: `https://aclanthology.org/L18-1551`

- This publication describes a dataset that can be used for training of summarization models of news in Czech language and related experiments.
- Citations: 22

Plátek, O. – Bělohlávek, P. – Hudeček, V. – Jurčíček, F. Recurrent neural networks for dialogue state tracking. *arXiv preprint arXiv:1606.08733*. 2016

- In this publication we introduce an RNN-based model used to dialogue state tracking as a sequence modeling task.
- Citations: 11

Schaub, L.-P. – Hudecek, V. – Stancl, D. – Dusek, O. – Paroubek, P. Définition et détection des incohérences du système dans les dialogues orientés tâche. (We present experiments on automatically detecting inconsistent behavior of task-oriented dialogue systems from the context). In *Actes de la 28e Conférence sur le Traitement Automatique des Langues Naturelles. Volume 1 : conférence principale*, p. 142–152, Lille, France, 6 2021. ATALA. Available at: `https://aclanthology.org/2021.jeptalnrecital-taln.13`

- This work discusses the inconsistencies and problems found in dialogue corpora
- Citations: 5

Mukherjee, S. – Hudeček, V. – Dušek, O. Polite Chatbot: A Text Style Transfer Application. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, p. 87–93, 2023

- In this work we explore the abilities of language models for text style transfer.
- Citations: 1

Plátek, O. – Hudeček, V. – Schmidtová, P. – Lango, M. – Dušek, O. Three Ways of Using Large Language Models to Evaluate Chat. *arXiv preprint arXiv:2308.06502*. 2023

- In this work we propose ways of evaluating dialogue qualities with large language models and describe our submission to the DSTC 11 challenge.
- Citations: 0