Dialogue Systems in a Low-resource Scenario

PhD. Thesis Proposal

Vojtěch Hudeček
Faculty of Mathematics and Physics, Charles University,
Malostranské Náměstí 25
118 00 Prague, Czech Republic

Abstract

Dialogue system research is a very active topic nowadays. However, deployed systems are usually very specifically focused and are not able to hold a complex conversation. One of the reasons is that there aren't a lot of methods that would allow us to extend a trained system in a simple way. Moreover, training a system usually requires a large amount of well-annotated training data. Data annotation is especially problematic for dialogue systems due to its relatively high abstraction and complexity. Therefore, we propose to invent new learning methods for dialogue systems which would lead to quality improvement and widen dialogue systems use cases. We plan to use unsupervised methods that bring us the possibility of using much larger unannotated corpora and hence more effective training of statistical models. We further focus on exploiting weekly annotated data using methods such as transfer learning or meta-learning (learning from similar tasks). These techniques would enable usage of partially annotated data for dialogue system training. These methods are still severely underexplored in the area of dialogue systems.

1 Introduction

Human language is a convenient and the most 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 in the same way as they would with other human individuals.

To achieve this goal, we need to be able to transfer the information between human users and the computer. Humans most often use speech or written text to encode and transfer the information and there are techniques 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 perform a meaningful dialogue, we need more than just to mimic the interface. The computer should be able to understand the meaning of utterances in the context and provide relevant responses. In this work, we focus on this part of the problem, i.e. we do not care about the process of 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) is to generate the correct natural language response r given the natural language user utterance u and context c. The dialogue is a *turn-taking* conversation, i.e. participants (user and system) communicate in alternating *turns*.

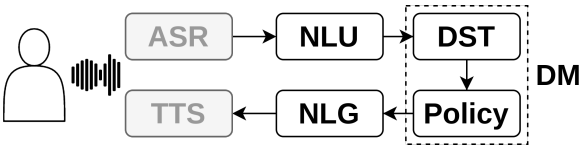

Figure 1: Overall architecture of task-oriented dialogue system pipeline. The data flow is outlined with arrows. ASR and TTS modules (depicted in gray) are not discussed in this work but are often included with the rest of the components.

The ultimate goal is to construct a dialogue agent that provides meaningful responses to all kinds of questions taking the conversation history into account. Such agent would effectively pass the Turing test, the holy grail of sort for the field of Artificial Intelligence. This goal is most likely far from being achieved, nevertheless, in many real life cases we don't need such complexity.

Dialogue systems promise a convenient means of communication between human 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 task have huge potential to automate tech support

processes, call centers or serve as personal assistants.

Despite some successful dialogue system deployments, dialogue systems still suffer from a number of drawbacks. Usually, the DSs are tailored to specific applications and it is hard to apply them in other domains. This results in bad scalability and inflexible applications. Another problem is that they require a lot of annotation for the training data, especially in the task-oriented setting. Last but not least, 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 this thesis, we aim to propose solutions to some of these problems, especially in the task-oriented setting. We now outline the main goals we want to achieve:

- Make the task-oriented dialogue systems more scalable and easier to extend.
- Utilize transfer learning so that domain adaptation is easier.
- Enable the dialogue systems to leverage large unannotated data sets and consequently train more robust models.

Scalability and domain adaptation goes hand in hand. We focus on reduction of the amount of annotation needed to train a system and on knowledge abstraction in order to make transfer learning possible. To be able to leverage larger data sets, we explore unsupervised techniques which do not require annotation and therefore make the data collection process substantially easier.

In the following section, we introduce the current state of task-oriented dialogue system research. Section 3 discusses the subject of dialogue system evaluation. Next, we introduce our research conducted so far and outline the future work in Sections 4 and 5 respectively. We conclude in Section 6.

2 Dialogue Systems Implementations

Because of varying use cases of DS, the architectures may vary a lot. We thus introduce a classification of dialogue systems that reflects the expected capabilities. There are multiple approaches to define 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, QA task can be seen as an instance of a simple conversation. The main task of a QA system is to provide answers to the user's questions. The topics may vary a lot and good understanding is essential for this task as well as knowledge representation. 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 that 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 database. Here the dialogues are usually much more complex than in the QA setting and dialogue context has to be taken into account.
3. **Chit-chat** - In some cases, we might be interested in a system that is able to talk to the user casually and provide entertainment. Such systems might be used in combination with task-oriented systems to serve as human-like virtual assistants or possibly use the dialogue to advertise for 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 are able to work only in one topic area, e.g. public transport or restaurant information, whereas **multi-domain** systems are able to handle multiple domains. These types of systems aren't able to give meaningful answers outside of the domains that they're trained on. A dialogue system is considered **open-domain** if it's able to 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, with internet access and smart information retrieval methods, the systems are able to cover tens of different domains.

Here we focus on the task-oriented DS and discuss it in more depth. From the domain perspective, task-oriented DS are usually either single or multi domain systems, open-domain is not often the case.

We can see a task-oriented dialogue as a slot filling task. That means we have a predefined set of semantic *slots* that need to 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). DA is a tuple consisting of user *intent* (overall meaning of the sentence) and optionally also *slot* and the corresponding *value*. In case that multiple slot values are present, all are considered to have the same intent. An example dialogue with respective DA representation is depicted in Table 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. Therefore, we focus on data-driven approaches based on machine learning models. In the following we introduce various approaches to dialogue system implementations. Section 2.1 overviews the traditional approaches to implement dialogue system as a modular pipeline. Next, Section 2.2 discusses the end-to-end models and Section 2.3 introduces unsupervised methods.

2.1 Modular 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. A typical pipeline is depicted in the Figure 1. First, the Natural Language Understanding (NLU) module parses the utterance and creates structured representation. Based on NLU outputs, the dialogue management module determines the next action. Dialogue Management usually consists of the state tracker that updates state based on NLU outputs and the policy module that chooses the action. Finaly, the language generation module is used to verbalize the chosen action.

2.1.1 Natural Language Understanding (NLU)

The purpose of NLU is to extract the meaning of input utterances in natural language and transform it to a structured representation, i.e. dialogue acts. Basically, the NLU module has three subtasks. It has to determine the domain of the utterance, detect the user intent and capture any slot values, if present. 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). Because of the similar nature of these three sub-tasks, it is reasonable to model them jointly. Especially 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).

2.1.2 Dialogue State Tracking (DST)

Dialogue state is used to keep track of the dialogue history, effectively providing the necessary context. Dialogue State Trackers are used to update the state with correct values after each turn. The most straightforward solution to this problem is a rule-based system that simply tracks the current slot values based on NLU. However, the situation is usually more complicated. We need to take into account a distribution of slot value probabilities and the update rules can be rather complex. Žilka et al. (2013) provides a comparison of different data driven models for dialogue state tracking. Neural networks have also been used to model the distributions (Mrkšić et al., 2016; Zhong et al., 2018) and deal with multiple domain handling (Rastogi et al., 2017).

2.1.3 Dialogue Policy

The core component of the DS is the dialogue policy. Its responsibility is to make the decision which action should the system take in each turn. 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).

2.1.4 Natural Language Generation (NLG)

When the decision on a system action is made, the system needs to verbalize the action. In other

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 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

words, we need to create an utterance in natural language that expresses the information given in the system's underlying representation. NLG is often realized with a set of handcrafted templates which are selected heuristically (Rudnicky et al., 1999). 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). Lately, neural network based systems were proposed as well (Wen et al., 2015, 2016)

2.2 End-to-end architectures

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 that were made by the modules early on in the pipeline. Another disadvantage is the way how these systems are trained. Each component requires specific data annotation, thus it can be difficult and costly to obtain a dataset suitable for training all of the components. Also, the system design itself is more complicated since it requires implementation of multiple models.

Various end-to-end solutions have been proposed to address the drawbacks of modular system training. This was made possible largely thanks to the growing popularity of Neural Networks (NN) and the backpropagation algorithm over the last decade. NN form a family of models that naturally allow us to combine multiple models and train them using single training algorithm. Therefore several solutions were proposed that implement the respective modules using neural network based models, interconnect them to form the pipeline and train them jointly (Li et al., 2017; Wen et al., 2017b). Although the end-to-end training improves 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 not suitable for practical use in task-oriented DS in its raw form due to overall low performance and insufficient robustness. The idea was further extended by Williams et al. (2017) who introduced the *Hybrid Code Networks*, an architecture that uses multiple utterance representations which are customizable by the developer. Despite good performance and flexibility, the proposed model again required a non-trivial amount of data annotation. Lei et al. (2018) came with 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 prior to response generation. They summarize the dialogue history in the RNN hidden state and use a system of copy mechanisms to be able to track the dialogue state. The proposed dialogue state representation is greatly simplified and doesn't require explicit NLU input, thus the annotation process is significantly easier.

In recent years, the NLP word has witnessed a great success of attention based models (Transformers) (Vaswani et al., 2017) and their usage as pre-trained language models (Devlin et al., 2019). In the area of 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 pretrained models are naturally utilizable for transfer learning, which proved to be useful in dialogue domain adaptation task (Shalyminov et al., 2019). Recently, attention-based architecture was proposed that models latent dialogue actions (Bao et al., 2019).

2.3 Unsupervised and transfer learning methods

The research of methods that reduce the amount of supervision needed can be divided into two paradigms. One direction of research 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 doesn't need supervision or requires less annotation.

2.3.1 Unsupervised analysis and labeling

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 convolutional neural network that is trained to score the sentence-intent similarities. Recently, Shi et al. (2018) proposed an intent detection model with the use of sentence clustering based on sentence-level features. They have applied their method successfully for the task of intent detection.

The idea of 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) which is a triple-based database of entity relations. 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.

Brychcín and Král (2016) focused on modeling the dialogue as 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. Shi et al. (2019) took this approach one step further by using more complex model based on RNNs and Variational Autoencoders (Section 4.2.1). After fitting the model, they analyse the hidden state transitions and infer the dialogue structure from it.

2.3.2 Modeling dialogues with less supervision

Work regarding the usage of 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 by definition structured and might be quite complex.

The method proposed by Jin et al. (2018) builds on Lei et al. (2018)'s sequence-to-sequence dialogue model (see Section 2.2) 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 in a semi-supervised way. For supervised data, both tracker modules are trained with supervised classification loss. For unsupervised data, teacher modeule 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 can't 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 (2018) approached the problem differently. They designed a novel dialogue system model based on VAEs (Section 4.2.1). 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 transfering to another domain, the model needs only a small number of so-called seed responses to adapt. Based on this idea, other works followed (Shalyminov et al., 2019; Huang et al., 2019).

3 Dialogue system evaluation

There are multiple different criteria for dialogue systems evaluation. In case of modular systems, the individual modules can be evaluated separately. For NLU, we usually use **F1-score** for slot tagging and classification **accuracy** for intent detection and domain classification. For dialogue state tracking, similar metrics can be used. It is common to measure **Joint Goal Accuracy**, which calculates the proportion of dialogue turns where all the user constraints (i.e., dialogue state summarizing slot

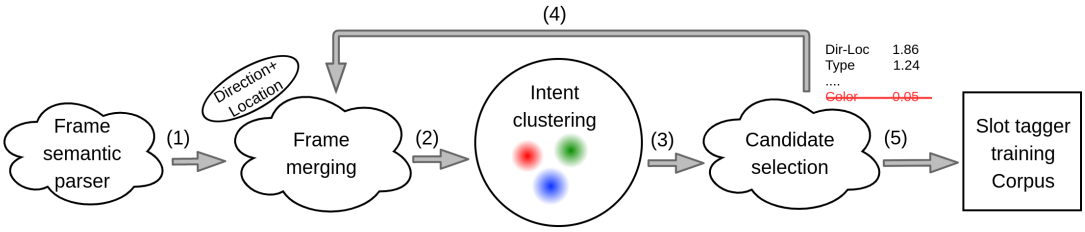

Figure 2: The process of obtaining labels for slot tagger training data. The utterances are first tagged with a semantic frame parser and frame features such as frequency and coherence are computed **(1)**. Frame similarities are analyzed so similar frames can be merged to form one slot candidate **(2)**. Utterances are then clustered according to their verbs and frame content **(3)**. Within each cluster, frames are scored and lowest-ranking frames are discarded **(4)**. Steps (**2-4**) are repeated until no more candidates are eliminated. Finally, we obtain a corpus labeled by the slot candidates and use it to train the slot tagger **(5)**.

values) are captured correctly (Mrkšić et al., 2016).

It is considerably harder to measure a system's overall performance. The policy module's decision is difficult to evaluate unless we have turn-level action annotations. Many use **BLEU** (Papineni et al., 2002) score, well-known from the area of machine translation. However, 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, again it is not straightforward how to define the dialogue success. Many systems use user simulators to allow employment of reinforcement learning techniques. In such scenarios, the user behavior is model based and can be nondeterministic, therefore defining success is challenging. Usually, it is based on evaluation of user and system dialogue acts, which relies on good and extensive data annotation.

Due to the mentioned challenges, **human evaluation** remains the best way to evaluate the DS. In our work, we use the above mentioned intrinsic evaluation measures (F1, joint goal accuracy). Although BLEU has its drawbacks, it's useful to compare different model variants, especially in combination with the other metrics. Due to the nature of our task, it will be necessary to employ human evaluators, possibly in combination with trained automated evaluation models (Lowe et al., 2017; Sarikaya et al., 2018).

4 Proposed approach and experiments

We focus on both artificial labeling and dialogue modeling, because the tasks are complementary in some aspects and can mutually benefit from each other. Our ultimate goal is to reduce the amount of supervision needed to train and deploy both modular and end-to-end dialogue systems. To tackle the issue of unsupervised annotation, we proposed a weakly-supervised technique for language understanding introduced in Section 4.1. This technique performs both slot induction and intent detection based on corpus analysis. We further focus on the dialogue end-to-end response generation task and introduce a novel modeling approach in Section 4.2. One of the key desired properties of the proposed model is its interpretability, which we discuss in Section 4.2.3.

4.1 Distant supervision for Language Understanding

We propose an iterative method which uses generic frame-semantic and semantic role parsers, a clustering algorithm, and a ranking model to perform intent detection and slot induction. The frame-semantic parser outputs are filtered and the slot structure is inferred, only the relevant concepts are captured. This method outputs artificial slot labels which we subsequently use to train a standalone slot tagger. Labels obtained with our method are shown in Table 2.

4.1.1 Method

The data are first labeled with FrameNet (Baker et al., 1998) semantic parser. FrameNet semantic frames are usually of much finer resolution than needed for a dialogue system, i.e. more than one frame usually refers to the same concept as a single dialogue slot. On the other hand, some of the frames recognized by FrameNet are completely irrelevant to the DS in the given dialogue domain. Therefore, we need to *merge* frames that map to the same slot and *eliminate* those which are irrelevant. In the following procedure, we attempt to identify

user input 1:	*I would like an expensive restaurant that serves Afghan food*
frame parser:	{Locale: restaurant, Expensiveness: expensive}
our slot tagger:	{slot-0: **Afghan**, slot-1: **expensive**}
user input 2:	*How about Asian oriental food?*
frame parser:	{Origin: Asian, Food: food}
our slot tagger:	{slot-0: **Asian**}
	...

Table 2: A sample section of a dialogue from CamRest676 data, with labels provided by the frame-semantic parser and our slot tagger (bottom). Note that although "Afghan" food is not in the frame parser output, our model was able to recognize it. In the second utterance, the slot value for slot-0 (corresponding to food type) changes and the change is successfully captured. This demonstrates the ability of our model to categorize entities (both "Afghan" and "Asian" relate to the same slot).

slots and slot-related intents (i. e. dialogue acts). It is reasonable to address these tasks jointly since slot frequencies generally differ across intents. The iterative process is captured in Figure 2. In each iteration, it (1) merges similar frames, (2) clusters frame occurrences in the data to detect different intents, (3) ranks frames' relevance for each detected intent, and (4) eliminates irrelevant frames. Once no more frames are eliminated, the procedure is terminated and we obtain slot candidates, which are used to train a slot tagger.

4.1.2 Frame representation and merging

During the merging process, frames are represented as mean word embeddings of frame fillers; we denote the embedded frame f_1 as $e(f_1)$. We measure similarity of frames f_1 and f_2 as:

$$sim(f_1, f_2) = sim_e(e(f_1), e(f_2)) + sim_{ctx}(f_1, f_2) \quad (1)$$

where $sim_e(f_1, f_2)$ is a cosine similarity of frames' embeddings and $sim_{ctx}(f_1, f_2)$ is a normalized number of occurrences of f_1 and f_2 with the same dependency relation. If the similarity exceeds a preset threshold T_{sim}, the frames are merged.

For a verb-frame pair (v, f), we compute its feature representation $feat_{v,f}$ as a sum of embeddings:

$$feat_{v,f} = e(v) + e(f) \quad (2)$$

We assume that the verb-frame pairs (e.g. "like-expensive") correspond to individual intents in the corpus (i.e., intents here are slot-dependent). Therefore, we can cluster the verb-frame pairs to find similarities. We can regard this clustering as unsupervised intent detection. Following Shi et al. (2018), we use hierarchical clustering of verb-frame pair feature vectors. This clustering also helps the frame relevancy ranking process, because some frames might only occur very few times in the data and therefore they wouldn't be ranked high if the frame ranking was run globally. Because we run the ranking locally for each intent cluster, we solve the issue of different frame occurrence frequencies.

To remove frames irrelevant for the dialogue domain, we rank the frames and remove the lowest-ranking ones. To rank the frames, we use frame frequency, mean pairwise similarity of fillers and TextRank algorithm (Mihalcea and Tarau, 2004). The final frame score is a simple sum of rankings with respect to all four features. We then choose the N top ranked candidates.

4.1.3 Slot Tagger Model Training

Our method can yield a good set of dialogue slots to be tracked. However, using the clustered and filtered frame parser annotation directly can result in low recall since the frame parsers are generic and not adaptable to our specific domain. Therefore, we use the noisy labels obtained by the parser and use it to train a new, domain-specific slot tagging model to improve performance. The model has no access to better labels than those derived from the frame parser; however, it has a simpler task, as the set of target labels is now much smaller and the domain is much narrower then when the original frame parser was trained.

We model the slot tagging task as B-I-O-coded sequence tagging, using a convolutional neural network that takes word- and character-based embeddings of the sequence tokens as the input, and it outputs a sequence of respective tags (Lample et al., 2016).[1] The output layer of the tagger network gives softmax probability distributions over possible tags. To further increase recall, we add a simple 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.

[1] https://github.com/deepmipt/ner

method	CamRest676	CarSLU	WOZ-hotel
Chen et al.	0.535 ± .002	0.590 ± .001	0.382 ± .001
Ours-nocl	0.301 ± .006	0.372 ± .011	0.121 ± .001
Ours-pars	0.541 ± .008	0.665 ± .007	0.384 ± .002
Ours-full	**0.663** ± .012	**0.687** ± .009	**0.543** ± .004

Table 3: F1 score values with 95% confidence intervals for unsupervised slot filling performance comparison among different methods). The measures are evaluated using a manually designed slot mapping to the datasets' annotation.

4.1.4 Results and discussion

We conducted a series of experiments to evaluate the candidate selection process and frame merging. Although the method is unsupervised, it is needed to construct handcrafted mapping of slot candidates to ground-truth slots.

We compare the intent detection accuracy to two baselines: (1) a method that trivially labels all utterances with the most common label (**majority**) and (2) a method that clusters the utterances based on average embedding vectors. We present the intent detection results in Table 4. For some of the datasets, **majority** is a strong baseline (*WOZ-hotel*), but the embedding clustering does not perform very well. Our experiments showed that the clustering yielded by our method is reasonable and mostly outperforms the majority baseline.

Perhaps more importantly, we evaluate the slot tagging results using F1 score. In Table 3, we provide the overview of slot tagging results on multiple selected datasets. **Ours-full** refers to our full method, while **Ours-nocl** and **Ours-pars** refer to modifications in which we don't use the clustering process or the neural tagger, respectively. We compare the results to a method previously proposed by Chen et al. (2015, see Section 2.3.1). Slot induction consists of two subtasks – we need to (1) determine the slots (i.e. what are the concepts we are interested in) and (2) tag them correctly. Therefore, it is rather complicated. We propose a method that can find reasonable slot candidates and yields a standalone tagger that is applicable without any additional dependencies. The slot induction performance is promising, the trained tagger proved to improve recall over the original parser. However, the drawback is that some types of concepts might not be captured by the frame-semantic tagger at all, therefore the method is not able to take them into account, not even with generalization provided by the trained tagger.

method	CamRest676	CarSLU	WOZ-hotel
Majority	0.592	0.530	**0.883**
Embeddings	0.535	0.551	0.873
Ours	**0.705**	**0.613**	0.882

Table 4: Intent detection accuracy of our methods if we interpret the clustering results as user intent detection. *Majority* is a majority baseline and *Embedding* refers to an average sentence embedding clustering approach.

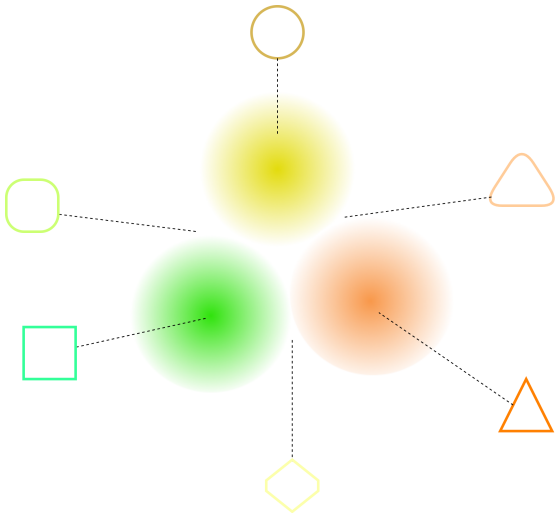

Figure 3: Variational autoencoder latent space. The encoder distributions are distinguished by colors and different classes by shapes. It can be seen, the interpolation between two sampled points is meaningful

4.2 Modeling the dialogue

We attempt to generate dialogue responses in an end-to-end fashion (see Section 2.2), using no dialogue state supervision, but attempting at interpretable internal dialogue states. Here, Shi et al. (2019) propose a novel usage of the VRNN model which we introduce in Section 4.2.1. They use the model to analyze the dialogue data and uncover the underlying structure. To achieve this, they use the concatenation of the user and system utterances as an input. Because of the system utterance usage, their approach is not suitable for dialogue generation. However, the VRNN model has great capabilities for dialogue modeling.

4.2.1 Background

Variational autoencoders Neural network training is a process during which the network learns to create internal representations of data in order to accomplish a given task. In 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 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 need to be made in order to obtain better representations. The technique proposed by Kingma and Welling (2013) 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 $\mathbf{z}$. VAEs impose prior distribution on the latent variable $\mathbf{z}$, which acts as a regularization during training and makes it possible to draw samples from q. Consequently, the VAE latent space is regular in a sense that it is possible to interpolate between two points. The latent space structure is depicted schematically in Figure 3. Typically, the modeled distributions are Gaussian and the prior is the standard normal distribution $N(0, 1)$.

We can realize the function 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, 2013). The reparameterization trick uses the fact that a random variable under certain conditional distribution can be expressed as a deterministic transformation of some other variable with independent marginal distribution. Distributions that allow us to do such a transformation include *Gaussian, Logistic* or *Gumbel*.

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 possibly uncover underlying processes in sequential tasks. It is problematic to incorporate discrete variables into neural network models, because the widely used backpropagation algorithm requires smooth differentiable functions in order 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 the Gumbel-softmax distribution (Jang et al., 2016) that enables us together with the reparameterization trick (Section 4.2.1) to work with categorical variables while not breaking the gradient flow in the network.

Variational Recurrent Neural Networks Variational Recurrent Neural Networks (VRNN) (Chung et al., 2015) explicitly model the dependencies between latent random variables across subsequent timesteps, similarly to dynamic Bayesian networks such as Hidden Markov Models (HMMs). Unlike HMMs, the transitions between latent states are dependent not only on the state in the current time step, but also on the RNN hidden state which allows to model more complex dynamics. Therefore, VRNNs are well suited for modeling sequential processes. Basically, the VRNN is a recurrent network with a VAE in every time step. The difference with respect to a basic VAE is that both prior and posterior distributions as well as the decoder are dependent on the RNN hidden state. Formally, let $\mathbf{h}_t$ be the RNN hidden state in time step t. The prior is realized with function $\varphi^{prior}(\mathbf{h}_{t-1})$, the posterior distribution q is modeled with $\varphi^{enc}(\mathbf{x}_t, \mathbf{h}_{t-1})$ and the generation decoder distribution is $\varphi^{dec}(\mathbf{z}_t, \mathbf{h}_{t-1})$.

4.2.2 Dialogue response generation with VRNN

End-to-end Dialogue generation task (see Section 2.2) can be described as finding the generation distribution of system utterances $\mathbf{s}$ given user utterance $\mathbf{u}$ and summary context $\mathbf{c}$. We propose to use the VRNN (Section 4.2.1) to model this task. The VRNN as we described it models a sequence of observations with one latent variable. To adjust the model for dialogue response generation, we use two VAEs in every time step, so we model the user and the system parts of the dialogue. We can regard the posterior module (network) of the VRNN as a teacher and the prior module as a student. During training, the student module learns to mimic the teacher distribution. When generating, the student distribution is used. The hidden state of the turn-level RNN then represents the context summarization, i.e. dialogue state. Both user and system VAEs depends on the hidden state, i.e. they share the same context representation. The system VAE is discrete and its prior module has additional dependency on the user latent variable $\mathbf{z}^u$. The generation story goes as follows: (1) The user pos-

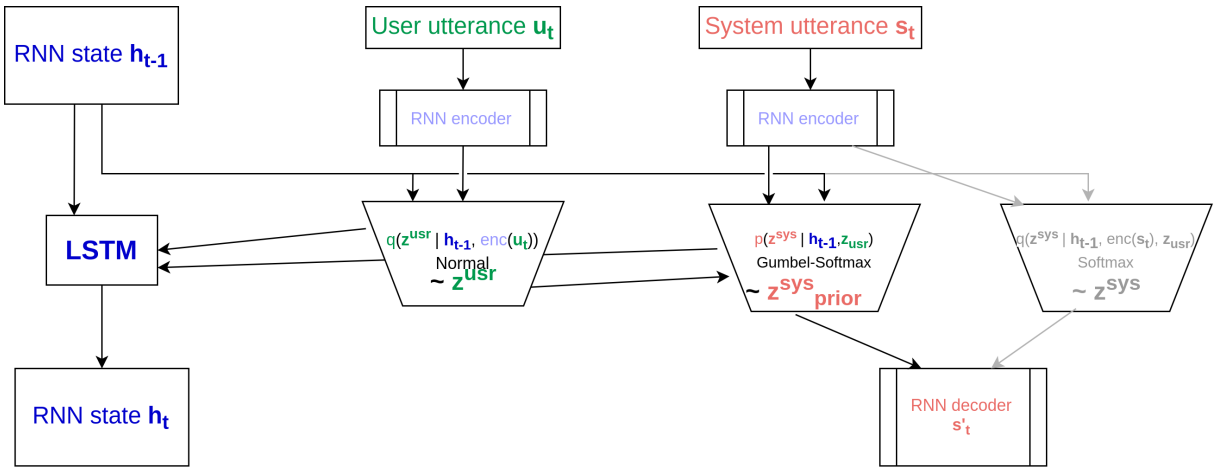

Figure 4: The simplified schema of our proposed VRRN model generation. The dependencies are depicted with arrows. LSTM module on the left corresponds to the turn-level RNN. The module is captured in generation time, therefore the posterior module is depicted in gray since it is not used. The user decoder is omitted for clarity.

terior module reads the input utterance and hidden state $\mathbf{h}$ and yields $\mathbf{z}^u$. (2) The system prior module yields $\mathbf{z}^s$ conditioned on $\mathbf{z}^u$ and $\mathbf{h}$. (3) The system decoder generates the response based on $\mathbf{z}^s$. (4) $\mathbf{h}$ is updated. The system VAE can be regarded as a policy module that operates with categorical variables – actions. The pipeline is schematically depicted in Figure 4.

There are different possibilities how to get utterance representations. The use of variational autoencoders is motivated by robustness of the representations and the possibility to work with categorical variables, as discussed in Section 4.2.1. The loss function is the timestep-wise variational lower bound. Let $p(\mathbf{z}^u_t|\mathbf{h}_{t-1})$ and $q(\mathbf{z}^u_t|\mathbf{u}_t, \mathbf{h}_{t-1})$ be the user prior and posterior latent variable distributions in time step t and $p(\mathbf{u}_t|\mathbf{z}^u_t, \mathbf{h}_{t-1})$ the user utterance generation distribution in time step t. For the system part, we denote the respective distributions analogically. Then the loss can be expressed as:[2]

$$\mathcal{L} = \sum_{t=1}^{T} (-KL(q(\mathbf{z}^u_t|\mathbf{u}_t)||p(\mathbf{z}^u_t)) + \log p(\mathbf{u}_t|\mathbf{z}^u_t)) + (-KL(q(\mathbf{z}^s_t|\mathbf{s}_t)||p(\mathbf{z}^s_t)) + \log p(\mathbf{s}_t|\mathbf{z}^s_t)$$

At inference time, we cannot condition on the future system response, therefore only the *prior* module is used.

4.2.3 Interpreting the latent variable

Because the model is forced to encode the system utterance using categorical variable, it might be seen as an encoded action. However, the actions will not likely be in a one-to-one correspondence to expert annotated actions. Therefore, a mapping needs to be introduced that helps to assign latent actions to their expert-defined counterparts. We propose to fit a decision tree that predicts the expert action from the latent vector (Figure 5). We can then extract a set of rules and construct the desired mapping, if required. The decision tree usage might seem too complicated for one-hot variables, nevertheless, it's justified in case more complex representations are used, e.g. multiple random variables.

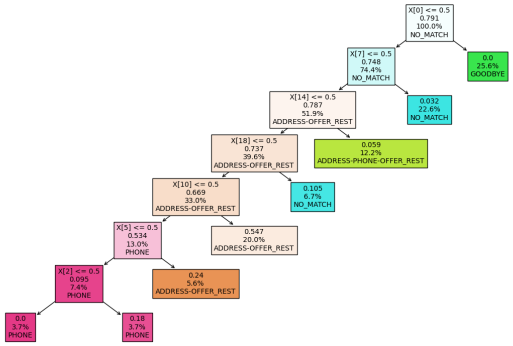

Figure 5: An example of decision tree that was built to help interpret the latent system action. Expert-defined actions are distinguished by different colors and we can see that the learned representations are meaningful and interpretable.

5 Future Work

In Section 4.2, we describe a novel usage of the VRNN model that incorporates discrete actions to

[2]We omit the RNN hidden states for readability.

achieve greater interpretability. Our main motivation is to design a system that can be used in the task-oriented setting. To achieve this, it is essential to track the dialogue state since it allows to communicate with databases or external APIs.

5.1 Immediate plans

We can extend the proposed VRNN architecture and make the latent variable representations more complex. Zhao et al. (2018) uses multiple latent vectors in VAEs. We plan to adopt this and add a new term to the loss function to make the autoencoder to save NLU or DST information in the latent vectors. The first step shall be teaching the model to recognize a domain in the latent variable. This approach allows for the use of semi-supervised learning, because we potentially need just a fraction of data annotated with NLU to help the system to extract the right information in a particular latent vector.

We also incorporate copy mechanism (Gu et al., 2016) to our model so it can track the slot values. However, preliminary experiments showed that the copy mechanism can be too strong and prevents the model from learning meaningful representations, similarly to vanishing latent variable problem (Bowman et al., 2015). Therefore we need to change the way the copy mechanism is integrated to mitigate this issue. If the model captures the NLU information successfully, we could also analyse the latent space and extract the knowledge. Hence, the trained model could be used as an annotation tool. We plan to employ the artificial labels obtained by our NLU tagger (Section 4.1) to help bootstrap the system.

5.2 Long term goals

Despite being trained from data in an unsupervised fashion, dialogue systems still need to be able to communicate with external knowledge sources, e.g. database or an external policy. Some minimal level of annotation will likely always be needed to model the task-oriented dialogue. Therefore, domain adaptation techniques are important to be able to reuse the knowledge obtained by the system in another domain. Our proposed VRNN-based model uses multiple discrete vectors to encode actions. Assuming the set of possible actions is much smaller than the number of possible representations, the model is redundant because it utilizes only a portion of the available vector space. We can use this property to our advantage since the model is effectively prepared to handle new actions. Also, some of the actions might be generic, e.g. *goodbye()*, therefore they can be reused in the multi-domain scenario without additional effort (Keizer et al., 2019). If trained correctly, the model could encode the actions using hierarchical semantics (one vector determines the action type, other determines the arguments) which further increases the transfer learning potential. Consider the actions '$ask\_time(train\_leaves)$' and '$ask\_time(bar\_opens)$'. If represented hierarchically, only the action argument has to be learned when switching the domain from trains to restaurants.

Following the recent trend, we want to exploit great modeling power of pretrained attention-based models. Some solutions have already been proposed to work with discrete variables using Transformer models (Bao et al., 2019). The ideas proposed in the previous paragraphs can be extended to work with these types of models and possibly improve the performance, mainly in terms of language generation.

Our weakly supervised slot induction method is currently in the review process and we aim to publish our VRNN model this year as well. After publishing, we focus on the hierarchical modelling of the actions and corresponding transfer learning capabilities. We assume one or two publications about this topic. Afterwards, the goal is to explore the usability of our method with Transformer models.

6 Conclusion

Dialogue modeling is an active topic with many subtasks. Despite well functioning applications, a lot of data-driven systems are dependent on multiple level of annotations which limit their scalability and flexibility. We address these issues in several ways:

1. We make the annotation process easier by proposing weakly supervised techniques for automatic annotation.

2. We propose a novel way of dialogue modeling that doesn't require an extensive amount of annotation and allows for better interpretability.

3. We focus on semantics of the proposed model's representations and utilize them for domain adaptation task.

The main expected contribution of our work is designing a task-oriented dialogue system that significantly reduces the amount of required data annotation. Moreover, such a system should be easy to adapt to multiple domains and could be used also as a data analysis tool that uncovers latent dialogue actions.

References

Heike Adel, Benjamin Roth, and Hinrich Schütze. 2016. Comparing convolutional neural networks to traditional models for slot filling. *arXiv preprint arXiv:1603.05157*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proc. COLING*, pages 86–90.

Siqi Bao, Huang He, Fan Wang, and Hua Wu. 2019. Plato: Pre-trained dialogue generation model with discrete latent variable. *arXiv preprint arXiv:1910.07931*.

Tim Berners-Lee, James Hendler, Ora Lassila, et al. 2001. The semantic web. *Scientific American*, 284(5):28–37.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Tomáš Brychcín and Pavel Král. 2016. Unsupervised dialogue act induction using gaussian mixtures. *arXiv preprint arXiv:1612.06572*.

Guan-Lin Chao and Ian Lane. 2019. Bert-dst: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. *arXiv preprint arXiv:1910.07931*.

Yun-Nung Chen, Dilek Hakkani-Tür, and Xiaodong He. 2016. Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *Proc. IEEE ICASSP*, pages 6045–6049.

Yun-Nung Chen, William Yang Wang, and Alexander Rudnicky. 2015. Jointly modeling inter-slot relations by random walk on knowledge graphs for unsupervised spoken language understanding. In *Proc. NAACL*, pages 619–629.

Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. 2014. Leveraging frame semantics and distributional semantics for unsupervised semantic slot induction in spoken dialogue systems. In *Proc. IEEE SLT*, pages 584–589.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988.

Mark G Core and James Allen. 1997. Coding dialogs with the damsl annotation scheme.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Milica Gašić, Filip Jurčíček, Simon Keizer, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. 2010. 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*, pages 201–204. Association for Computational Linguistics.

Milica Gašić and Steve Young. 2013. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Larry Heck and Dilek Hakkani-Tür. 2012. Exploiting the semantic web for unsupervised spoken language understanding. In *Proc. IEEE SLT*, pages 228–233.

Xinting Huang, Jianzhong Qi, Yu Sun, and Rui Zhang. 2019. Mala: Cross-domain dialogue generation with action learning. *arXiv preprint arXiv:1912.08442*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Xisen Jin, Wenqiang Lei, Zhaochun Ren, Hongshen Chen, Shangsong Liang, Yihong Zhao, and Dawei Yin. 2018. Explicit state tracking with semi-supervision for neural dialogue generation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1403–1412. ACM.

Dan Jurafsky. 2000. *Speech & language processing*. Pearson Education India.

Simon Keizer, Ondřej Dušek, Xingkun Liu, and Verena Rieser. 2019. User evaluation of a multi-dimensional statistical dialogue system. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 392–398, Stockholm, Sweden. Association for Computational Linguistics.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. NAACL*.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. 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)*, pages 1437–1447.

Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. 2017. End-to-end task-completion neural dialogue systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 733–743, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126.

François Mairesse and Steve Young. 2014. Stochastic language generation in dialogue using factored language models. *Computational Linguistics*, 40(4):763–799.

Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. 2014. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proc. EMNLP*.

Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2016. Neural belief tracker: Data-driven dialogue state tracking. *arXiv preprint arXiv:1606.03777*.

Alice Oh and Alexander Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *ANLP-NAACL 2000 Workshop: Conversational Systems*.

Aaron van den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. *arXiv preprint arXiv:1704.03084*.

Abhinav Rastogi, Dilek Hakkani-Tür, and Larry Heck. 2017. Scalable multi-domain dialogue state tracking. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 561–568. IEEE.

Alexander I. Rudnicky, Eric H. Thayer, Paul C. Constantinides, Chris Tchou, R. Shern, Kevin A. Lenzo, Wei Xu, and Alice Oh. 1999. Creating natural dialogs in the Carnegie Mellon Communicator system. In *Proceedings of the 6th European Conference on Speech Communication and Technology*, pages 1531–1534.

Ruhi Sarikaya, Daniel Boies, Paul A Crook, and Jean-Philippe Robichaud. 2018. Dialogue evaluation via multiple hypothesis ranking. US Patent 10,162,813.

Robert E Schapire and Yoram Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Igor Shalyminov, Sungjin Lee, Arash Eshghi, and Oliver Lemon. 2019. Few-shot dialogue generation without annotated data: A transfer learning approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 32–39, Stockholm, Sweden. Association for Computational Linguistics.

Chen Shi, Qi Chen, Lei Sha, Sujian Li, Xu Sun, Houfeng Wang, and Lintao Zhang. 2018. Auto-Dialabel: Labeling dialogue data with unsupervised learning. In *Proc. EMNLP*, pages 684–689.

Weiyan Shi, Tiancheng Zhao, and Zhou Yu. 2019. Unsupervised dialog structure learning. *arXiv preprint arXiv:1904.03736*.

Yangyang Shi, Kaisheng Yao, Hu Chen, Dong Yu, Yi-Cheng Pan, and Mei-Yuh Hwang. 2016. 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*, pages 393–399.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. On-line active reward learning for policy optimisation in spoken dialogue systems. *arXiv preprint arXiv:1605.07669*.

Dinoj Surendran and Gina-Anne Levow. 2006. Dialog act tagging with support vector machines and hidden markov models. In *Ninth International Conference on Spoken Language Processing*.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. 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*, pages 120–129, San Diego, California. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017a. Latent intention dialogue models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3732–3741. JMLR. org.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017b. 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*, pages 438–449, Valencia, Spain. Association for Computational Linguistics.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*.

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

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

Sibel Yaman, Li Deng, Dong Yu, Ye-Yi Wang, and Alex Acero. 2008. An integrative and discriminative technique for spoken utterance classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(6):1207–1214.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 35–45.

Tiancheng Zhao and Maxine Eskenazi. 2018. Zero-shot dialog generation with cross-domain latent actions. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 1–10, Melbourne, Australia. Association for Computational Linguistics.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. *arXiv preprint arXiv:1804.08069*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive dialogue state tracker. *arXiv preprint arXiv:1805.09655*.

Lukáš Žilka, David Marek, Matěj Korvas, and Filip Jurcicek. 2013. Comparison of bayesian discriminative and generative models for dialogue state tracking. In *Proceedings of the SIGDIAL 2013 Conference*, pages 452–456.