Title
Attribute Representation in Neural Language Models

Author
Yu, Dian

Publication Date
2022

Peer reviewed|Thesis/dissertation

Attribute Representation in Neural Language Models

By

Dian Yu

Dissertation

Submitted in partial satisfaction of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Office of Graduate Studies

of the

University of California

Davis

Approved:

Kenji Sagae, Chair

Zhou Yu

Premkumar Devanbu

Committee in Charge

2022

Contents

Abstract

Attribute Representation in Neural Language Models

Neural models, including neural language models and encoder-decoder models, are the backbone of current natural language processing (NLP) research. Large pre-trained models have greatly improved the performance of both language understanding and generation in many NLP tasks. However, information encoded from the pre-trained models cannot translate to target space easily, which typically requires fine-tuning on domain-specific tasks. Due to domain shift between the pre-training data and the task data, it is still challenging for models to adapt to downstream tasks, especially when the training examples are limited such as in few-shot and zero-shot settings. More importantly, the fine-tuned models can only work well for a small number of domains because of diverging from the original pre-trained model, thus are prone to have over-fitting problems with inductive bias. Although scaled up models with billions or trillions of parameters have shown promising performance with prompts and examples, the challenges still remain.

In this thesis, we study if we can learn and inject attribute representation to pre-trained neural models to solve the challenges. Different from a black-box model where the parameters contain vast but encrypted world knowledge, the learned attribute representation can guide the model to learn information relevant to the target task, or serve as supplementary information aside from the original parameters. Attributes can be as high level as language representation in a multilingual transfer learning setting, or as low level such as span or ontology representations. This direction is appealing since we can introduce new attributes to pre-trained models without requiring any changes to the original trained model parameters. As we will show, learning attribute representation is efficient in training with both computation and data requirements. Moreover, it is easy to do transfer learning with even only few examples, while maintaining the original model quality. We believe that training attribute representation is a critical step to reduce the gap between neural model pre-training and applying to target tasks.

Specifically, we first introduce methods to represent high-level attributes. Those

learned attributes can differentiate from other similar attributes so that they can be utilized to transfer useful knowledge across domains and further to control a neural model towards certain understanding and generation directions. Then, we discuss how to represent low-level attributes from pre-trained models. Those attributes can be hidden with pre-trained models and presented by latent representation. The representation can either be used directly for target tasks by identifying significant features, or be incorporated for further model training. Next, apart from more concrete attributes, we propose methods to integrate task specifications for efficient modeling. Those task-specific attributes model the target task directly, bridging model representation and prediction goals precisely, and enabling performance close to or even above human capacity. Lastly, we apply these attribute representations to dialog systems as a case study. We demonstrate how we can represent different aspects of attributes to build a dialog system from scratch smoothly. We present solutions to the most critical challenges in neural language models in general.

Acknowledgments

I got asked why I decided to do a PhD in NLP a few weeks before this thesis. When I look back from fives years ago, my answer is quite different now. I got a taste of research work in several labs in my undergrad study, and I was amazed by how much NLP could accomplish even before all the powerful pre-training. However, the way I think about doing research dramatically changed especially since I got the chance to explore different areas of NLP during my graduate school years. I underestimated, and also overestimated some of the challenges, but I am very grateful to the suggestions I received along the way.

Firstly, I would like to thank my advisors, Kenji Sagae and Zhou Yu. I did not have a clear research direction in mind when I started, but Kenji always believes in me and encourages me to explore even vague ideas as long as I find them interesting. Although many of the ideas proved to be not working, they really shaped the way I learn new methods, which is one of the most important skills I acquired in the past few years. I will always remember the suggestions he gave me for research and outside of research, as well as the random but delightful conversations in the lab. Meanwhile, Zhou trusted me in leading our team to build dialog systems where I got exposed to a wide range of NLP problems early on. She guided me to focus on challenging problems and has been inspiring my research vision. I am very fortunate to receive mentorship from advisors of different styles.

Secondly, I really appreciate the help from my mentors and my peers. I had my three wonderful internships at Uber AI and Google Research hosted by Yuan Cao, Luheng He, Chandra Khatri, Alex Papangelis, Mingqiu Wang, and Yuan Zhang. They, among with other senior researchers in the community, offered me opportunities to explore broader research directions and taught me to think bigger in my research scope. They made me better at taking research angles, and demonstrating my ideas and findings. I would also like to thank my peers Sam Davidson, Andrea Madotto, Quan Vuong, Qingyang Wu, Mingyang Zhou, and many more over the years. I could always get valuable suggestions from them either on my research problems, or any other things. They made everything simpler for me.

Last but not least, I want to thank my parents Shengli Yu and Wei Ma. They taught me to be persistent. I always have their support no matter what I decide to do, even the craziest things. They made my journey much easier especially during the Covid lockdown years. Their suggestions, even if not related to my work directly, often ended up benefiting my research.

I thank all the people who made the challenging PhD years enjoyable.

Chapter 1
Introduction

Learning dense representation for words started the era of pre-training for natural language processing [1, 2, 3]. Since then, from independent word embeddings to contextualized token representations, pre-training in a self-supervised way on a huge amount of data [4, 5, 6, 7] become the natural starting point for most NLP applications. These achievements enable products such as automatic translation and virtual personal assistant, and make people's lives more convenient without relying on experts.

Typically neural language models are trained to predict some words in a sentence to utilize massive training corpus such as Wikipedia and Reddit without requiring any human supervision. At this point, the pre-trained models can either be used to fill in some blanks, or complete some sentences freely. To accomplish downstream tasks ranging from text classification to reading comprehension, and more recently to open-book multi-hop question answering, engaging dialog systems, and towards reasoning, the pre-trained models need to learn how to translate what it already knows to the target space. The traditional solution to resolve this mismatch is to fine-tune on additional collected in-domain supervised data, so that we can change the model representation tailored to the target labels. Despite recent efforts in engineering prompts to enable large-scale language models to perform reasonable given a handful of examples, adapting pre-trained models to target tasks across domains is still demanding, especially for tasks that require additional knowledge or have complex target space. The central thesis of this work is thus

to answer the question whether we can map the pre-trained representations to the target representations more efficiently.

In addition to the challenges in pre-training and fine-tuning due to the differences in tasks, there is also a mismatch in domain differences. Moreover, although the models trained by predicting words exhibit strong understanding and generation ability demonstrated by probing and prompting research, we do not have any control of the model. In other words, when condensing world knowledge into dense representation, it is intriguing what the model knows, and how much it needs to learn to finish specific tasks rather than guessing words correctly. Importantly, once the model is pre-trained, the knowledge is kept frozen up to a certain time period, indicating that unless retrain with updated information with tremendous cost, there would be serious complications such as hallucination and toxicity. Motivated by the question of controlling pre-trained neural models, we explore directions to learn attribute representations that guide the neural models towards target tasks. This will make either a modularized or an end-to-end model controllable for both understanding and generation perspectives.

1.1 Modularized to End-to-End Models

Conventionally, modularized models are employed by a pipeline of natural language understanding (NLU) tasks such as part-of-speech tagging, coreference resolution, and parsing to generate symbolic representation before finishing the ultimate tasks such as question answering. There are two main benefits for such a pipeline. Firstly, rather than completely a black box, the models are more controllable. We can observe what all the intermediate representations are, and how each step influences the target output. Let us take dialog systems as an example, with the clear separation for NLU, dialog manager, and NLG. If any response generated is not expected, we can trace back to see if there is any error with the NLU part such as a wrong intent detection, or a wrong policy planned. Then it becomes easy to fix these problems and make the model more precise. Secondly, compared to the downstream specific use cases and target spaces, modularized models usually have simpler target space such as multi-label classification. This makes transfer

learning more accessible since there are similar tasks, and makes human annotation and data augmentation more straightforward. These two are important considering factors for real-world products.

In spite of their benefits, modularized models are not robust, especially when new features require different annotation for intermediate representations. For example, for a new domain, not only do we need parallel data between inputs and target outputs, we also need a new set of intermediate representations and symbols. In comparison, end-to-end models take the original texts as input and output the target sequence, such as the answer to a question in a paragraph, directly. Combined with the powerful pre-training, end-to-end models are appealing, especially because they are more generalizable to unseen scenarios.

Regardless of the tempting benefits, one main caveat of end-to-end models is that most of the time they are black-box models. In other words, we can only observe what the model decides to output, but have no idea what its "thinking process" is, i.e., how a decision about whether a label, or a token, is made. There are many efforts trying to probe what the pre-trained models learned and found that intermediate results from modularized tasks are actually inherently modeled. Yet, we don't have any control, and cannot really inspect if, for example, the intent is correctly recognized by the model. Moreover, machine learning methods are known to be data intensive so that a large amount of task-specific data annotation is required to guarantee performance in the target task.

Attribute representation is a way to make the best of the two world. When applied on either a modularized model or an end-to-end model, attribute representation can make the model more controllable, while being more robust to changes and new requirements at the same time.

1.2 Omnipotent Large Neural Models

With recent success in both language and image pre-training using the transformer architecture, it seems that attention, if not "is all you need", is a potential framework to utilize different modalities, and maybe is indeed something important in scale-up modeling [8].

Based on the success in easy transfer learning from pre-training to downstream task, or prompt-based NLP methods with only few demonstrations [9] or webpage search [10], or even able to explain jokes [11], a natural question to ask is "is language model all we need[1]"?

On the one hand, the powerful ability to complete various tasks make some people speculate whether large-scale neural models are conscious and can lead the community towards artificial general intelligence[2]. On the other hand, the seemingly omnipotent models are more likely to simply reflect on whatever data they are trained with, while not considering critical characteristics such as intents and pragmatics [12]. This makes those models pointless without further supervised learning by either engineering prompts or examples with high variance, or by fine-tuning model parameters with a high cost. In addition to being data intensive, neural models are not generalizable to new tasks and cannot be easily customized to specific use cases. For example, it is relatively easy to build a general purpose translation system, but it is not trivial to translate texts that require domain knowledge such as scientific papers. Furthermore, although larger models perform better in different tasks, it is not clear what the model learns in its parameters and why the model makes a certain prediction. Although probing tasks have shown that the large models are knowledgeable, there is still a large mismatch between a general model and what we need, since we cannot squeeze the knowledge in a specific way to draw the connection. Therefore, those large neural models are largely limited from numerous perspectives.

Motivated by the current problems in terms of both model capacity and computation cost and corresponding impact, we propose to study attribute representation, which can potentially bring automatic prediction closer to human annotation in both inference resemblance and accuracy.

[1]Large language models (or encoder-decoder neural models in general in this context) can be applied to both modularized models as well as end-to-end models.

[2]`https://twitter.com/ilyasut/status/1491554478243258368?s=20&t=klsduOygu6srcG3i9QlfMw`

1.3 Motivation

Given by the limitations of large neural language models, we are interested in learning methods to translate from the general representation, or representation learned from different tasks, to the target task efficiently. We propose to leverage attribute representations and inject the learned representations into large neural models to control both modularized and end-to-end models, and turn the black-box into white-box models. This thesis demonstrates the idea by answering the following three research questions.

1. How to control a pre-trained language model for NLU and NLG tasks? Because the pre-training phase of self-supervised training with massive data do not consider any extra information, the first critical challenge is to control a neural model towards specific tasks. We introduce attribute representation methods in Chapter 3. Instead of the typical fine-tuning setup, we propose to connect the two phases by learning and aligning representations that the original model can recognize, while freezing the model parameters. We explain how this method can represent high-level attributes such as a specific language, or sentiments and topics, and why this is effective in controlling a neural language model in both understanding and generation tasks.

2. How to represent fine-grained attributes from neural models? Different from high-level attributes, low-level nuanced attributes are less straightforward to represent because they are usually not well defined and less distinct from each other. Although less studied than high-level attributes, low-level attributes are critical to represent features implicitly required for target tasks. We demonstrate how to learn and represent low-level attributes by pre-training and interpreting models for schema induction in the task-oriented dialog domain. Specifically, we introduce methods to construct the ontology of a task through detecting phrases and groups from in-domain pre-trained neural models.

3. How to align what a neural model already knows to what we need? High-level and low-level attribute representations serve as an intermediate step to connect learned parameters to the target tasks. However, there are tasks where attributes cannot be easily defined to draw the connection. Instead, representing task-specific attributes may be more straightforward. We introduce task-specific attribute representations from two

angles, to represent the target space by disentangling from the input representation, as well as to represent intrinsic latent structure. We illustrate the former from a retrieval-based few-shot metric-learning methods for both classification and structured predictions tasks, and the latter by representing systematic issues that trigger a pre-trained neural dialog model into generating problematic responses.

1.4 Outline

In the rest of this thesis, I will briefly introduce relevant background including how state-of-the-art neural language models function in Chapter 2, which are the backbone and prerequisite for attribute representation. In Chapter 3, we introduce high-level attribute representation, and we introduce low-level attribute representations in Chapter 4. Chapter 5 discusses how to represent attributes for task-specific features. In Chapter 6, we combine different attribute representation methods and apply them to building a dialog system as a case study. Finally, we summarize this thesis and analyze the key findings and address potential future directions in Chapter 7.

Chapter 2
Background

2.1 Overview

This Chapter briefly introduces the background for attribute representation. We first go through the widely used architectures including recurrent neural networks and Transformers, as well as their potential drawbacks and fixes. Then we review methods to pre-train large language models, and the potential problems that associate with this process. Finally, we expand the scope of language representation to multi-modal representation, as attribute representation can be utilized beyond language to other modalities such as images, although the main focus of this thesis is language.

2.2 Model Architecture

For models and experiments in this proposal, we mainly use two neural architectures, Long short-term memory (LSTM, [13]), and Transformers [14].

2.2.1 Recuent Neural Networks

Different from images where each pixel is continues and rendered by surrounding pixels, the nature of language considers each input as a linear sequence of discrete tokens, which may be considered as a Monte Carlo process, or take the whole context into consideration. To incorporate longer sequence without losing information, recurrent neural networks (RNNs) are introduced to process one token at a time while itertively consider previous

information. At each timestep t in a sequence of tokens $x_1, x_2, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots$, we consider both the current token x_t and previous hidden state h_{t-1} to calculate the current hidden state $h_t = a_1(W_{hh}h_{t-1} + W_{hx}x_t + b_h)$ and current output $y_t = a_2(W_{yh}h_t + b_y)$ where $W_{hh}, W_{hx}, W_{yh}, b_h, b_y$ are trainable parameters and a_1, a_2 are activation functions such as *tanh*. The hidden states are designed to capture all information, or in other words, to compress the past sequence into some fixed hidden representation up to a certain time step so that we can still model the sequence in a Monte Carlo fashion, without losing all the information about the sequence. Outputs are the actual observable representations that can later be used. When calculating the hidden states as well as outputs, the parameters we train can be considered as feed-forward neural networks (FF), and the activation functions aim to model nonlinear functions to have stronger representation power. Task-specific FF layers are common to map the RNN representations to target labels.

To mitigate the vanishing gradient problems in RNNs which makes it difficult to capture long context, LSTM learns gates to control how much information from the memory should be propagated to the current timestep and how much more recent information should be considered. In specific, LSTM is implemented with the following equation

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\
\tilde{c_t} &= tanh(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c_t} \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
h_t &= o_t \odot tanh(c_t)
\end{aligned}
$$

where i, f, c and o are input gate, forget gate, cell vectors, and output gate. The gates control how much information we should flow into the next hidden state, and how much we should just ignore since they may not be important to consider. W and U are learned weight matrices, σ is the logistic sigmoid function, and $\odot$ represents element-wise multiplication.

Similar methods such as Gate Recurrent Unit (GRU, [15]) were also introduced. We refer interested readers to their original papers for close comparison.

2.2.2 Transformers

RNN-based architectures are intuitive: we control what we need to memorize from the past, while considering new information at each new time step. However, because they can process long context by keeping a fixed representation as the history, this mechanism loses a lot of representation power, especially for contexts from many tokens before. Moreover, because of the iterative sequential order, encoding the input is relatively slow, particular when there are multiple layers.

In Transformers models, sequential dependencies are implemented with positional encoding as well as attention among token representations in multiple layers with layer normalization. Specifically, we learn independent weight matrices to map a word embedding or the representation from the previous layer (X) to key(K), value(V), and query(Q) representations for all the tokens in a sequence. Thus we can get contextual token representation through self-attention, which basically means to attend each token to the input itself, as

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

where d_k is the dimension of queries and keys. In this way, each token representation is weighted by how important other tokens (keys) are regarding the query (target token). It is therefore considered as contextual representation compared to word embeddings used directly in RNNs. Moreover, we can employ multiple heads where each head has its own set of key, query, value weight metrics to be able to encode the context into different views. Instead of iterative modeling, as we can see here, we can calculate all the contextual representation simultaneously of all heads. The only linear requirement is the number of layers, which is constant to the number of tokens in a sequence, making it very attractive for large model pre-training.

After the self-attention layer, we can learn some layer normalization to normalize the representation and speed up training. Meanwhile, this can also make the training at each

layer more independent because higher layers are not dependent on the scales of the lower layers. Different from batch normalization which normalizes the representation of each mini-batch, layer normalization is designed to better fit for language tasks where each example has different length and we may only be able to use small batch sizes due to the long sequence lengths, which make batch normalization not really feasible. Instead, we apply normalization on different dimensions. Specifically,

$$\mu = \frac{1}{d}\sum_{j=1}^{d} z_j$$

$$\sigma^2 = \frac{1}{d}\sum_{j=1}^{d}(z_j - \mu)^2$$

$$z_{norm} = \frac{z - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$\tilde{z}_{norm} = \gamma \cdot z_{norm} + \beta$$

where γ and β are learnable. In practice, we apply layer norm together with residual connection by $Z = LayerNorm(Z) + X$ before sending the Z into FF networks.

In an encoder-decoder architecture, in addition to self-attention in the encoder part, we can also apply cross-attention so that we consider self-attention on the already generated tokens, as well as the encoder representations.

Although Transformer architectures are powerful and efficient, one potential drawback is its quadratic computation cost since at each layer, we need n^2 computation of self-attention for a sequence of length n. Variations to solve this problem include calculating self-attention on certain tokens through potentially learnable patterns [16, 17, 18], employing global and local memory [19, 20, 21], and combining recurrence into Transformers [22, 23]. We refer readers to [24] for more details.

2.3 Language Models

Before large language model pre-training, previous research learns word embeddings with large dataset such as word2vec [2] which uses skip-gram with negative sampling to predict surrounding words, and GloVe [25] which also considers words' probability of co-occurrence.

In order to consider context in word embeddings, there are two major language model pre-training methods, causal language modeling and masked language modeling. We introduce the most popular language model pre-training methods here based on Transformers, as it is the backbone of recent pre-training models. We include sequence-to-sequence model here for simplicity in notation.

2.3.1 Causal Language Modeling

In causal language modeling, the goal is to predict the next token based on the already generated context, which is similar to recurrent neural networks. Specially, we are modeling $p(x_t|x_1, x_2, ..., x_{t-1})$. During pre-training, because we are not supposed to glimpse the future tokens, we need to apply some mask on the self-attention by only using the lower triangular of the attention matrix. At each time step, the model predicts some distribution as which token should be the next, and compare that to the ground-truth using loss functions such as cross-entropy.

Many powerful large scale models adapt this pre-training method such as GPT-3 [26, 27, 9]. Because the goal is to generate a sentence token by token, models pre-trained this way exhibit strong generation ability. It can also be applied to different generation scenarios such as dialog [28, 29, 30]. In addition, we can feed the model with some examples as the prompt, and ask the models to generate the target results. Although the pre-trained models are powerful, as motivated earlier, given some prompt, we have no information on what direction the model will generate.

2.3.2 Masked Language Modeling

Different from the auto-regressive nature of causal language model, the goal of masked language modeling is to utilize the bi-directional encoding ability. This has two benefits, namely better encoding ability, and faster encoding time since we do not require token-by-token encoding.

The masked language modeling task is to replace a portion of subwords (such as byte pair encoding [31] which tokenize words with frequently occurring subwords) in a sentence with a unique mask token or a random token and the training objective is to predict the

masked token distribution. Similar to causal language modeling, we compare the distributing to its ground-truth for training. Specifically, it models $p(w_t|w_1, ..., w_{t-1}, w_{t+1}, ...)$ where w_t is masked. It has been applied to BERT [7], XLM [32], and other models. For example, BERT encodes token embeddings, segment embeddings, and position embeddings to optimize the next sentence prediction (NSP) task which predicts if two sentences are consecutive, in addition to the masked language modeling task during pre-training. In comparison, XLM introduces a translation language modeling task which learns attention across language pairs with language embeddings associated with each token instead of NSP for multi-lingual language modeling. For downstream tasks, the pre-trained models fine-tune Transformer parameters on a relatively small annotated dataset together with some task-specific parameters such as linear layers on top of the last Transformer output.

Compared to causal language modeling, masked language modeling show better natural language understanding ability, probably because of the bi-directional encoding [7, 33, 32]. We still need to translate whatever the model parameters contain, to a target label space, which is not trivial.

2.3.3 Sequence-to-Sequence Model

Sequence-to-Sequence (Seq2Seq) models utilizes an encoder-decoder architecture [34]. The task for the encoder is to encode the input into some representations, and the task for the decoder is to generate a sequence of tokens based on the input. With the Transformers architecture, the encoder is always a bi-directional model with dense attention, while the decoder is a causal language model by both cross-attention on the encoder outputs, and the self-attention on generated tokens.

Pre-trained Seq2Seq models can leverage the powerful and efficient encoder similar to masked language modeling, as well as the generation ability [35, 36, 37]. Meanwhile, it also has the drawbacks for the previous two methods, namely not controllable, and requiring mapping to the target space.

The models mentioned above are mostly dense attention models. Recently improvement to utilize more parameters using sparse Transformers by activating small portions of the weights in a mixture-of-experts style [38, 39, 40] is another direction in scaling up

large neural language models.

2.4 Multi-modal Neural Model

In addition to language models where the input and the output are both texts, similar architectures can incorporate different modalities such as image and speech [41, 42, 8, 43, 44, 45, 46]. The main goal then is to align representations from different modalities into the same space, so that for example, the word "apple" would have very similar representation with an image of apples. Although powerful in different modalities through pre-training, we still cannot represent attributes that are relevant to scene images, or speech directly, and we cannot understand how the alignment happens between text and images for example in a specific task. This is not the main focus of this thesis, but attribute representations above language follow the same philosophy [47] and we point interested readers to relevant research for details.

Chapter 3
High Level Attribute Representation

3.1 Overview

In this chapter, we introduce methods to represent high-level attributes. We start with the general methodology for attribute representation, and then explain how this can be applied to sentiment and topic representation for language generation, as well as how to represent a slightly different but more abstract language representation for cross-lingual transfer learning.

This chapter is based on our works [48, 49] that were published at the ACL 2021 and EMNLP 2021 conference, which I lead as the main author. Taiqi He was a co-author for the language representation paper, and Zhou Yu and Kenji Sagae serve in an advisory capacity.

3.2 Introduction

While large pre-trained language models (LM) have advanced text generation with coherent language by training on a large amount of unlabeled data [26, 50, 36], they are not controllable. For instance, given the prompt "The issue focused on", GPT-2 [27] can generate a high-quality sentence, but it cannot take extra input such as "positive" or "business" to guide the sentence towards a positive sentiment or business-related topic, due to the lack of attribute labels during training. Similarly, despite recent efforts showing that pre-trained (multi-lingual) models can perform well across various languages, we do

Attribute	Generated Text
None	The issue focused on a 2008 decision by the United States Court of Appeals for the Ninth Circuit, in San Francisco, that denied local restaurants advance notice of changes to their menus, even when that change had not been submitted to ...
positive	The issue focused on returning to the simple premise that dialogue is more effective than banal reactions. They demonstrate very good personal style with establishing dialogue and bringing about a good point of view. Most fantastic of all ...
negative	The issue focused on a false belief that treatment can never be "good enough" and that long-term treatment only "cures" a person. This does not account for why this is the case: Patients with the ...
business	The issue focused on the regulations preventing banks and other entities in the financial sector from moving money across foreign borders without the consent of its investors.
athlete	The issue focused on Robinson, who went to camp with his hometown team after being released by the Seattle Seahawks, though it was ruled an emergency by the National Football League.
military	The issue focused on whether servicemen and women should be allowed to opt out of serving overseas. It was also about whether making it easier for American troops to return home would help their families.

Table 3.1. Examples generated using the proposed alignment function with Bayes disentanglement (`ACB`). Tokens underscored are the prompts. “None" indicates non-controlled generation from the original GPT-2 model. “business" is from AG News corpus, “athlete" is from DBpedia corpus, and “military" is not in the training data (zero-shot).

not have any control on how to utilize similar representation to transfer useful knowledge since similar languages can benefit from sharing parameters, but less similar languages do not help [51, 52].

To solve the discrepancy between training and inference, one direction is to train an LM from scratch with some supervision such as control codes in CTRL [53]. Nevertheless, this method requires training an LM with a large number of parameters, and is limited by the attributes used during pre-training. Another direction is to fine-tune the pre-trained LM on some annotated datasets. This usually requires updating all the parameters in the model, which incurs large computational costs with current large LMs that have

millions or billions of parameters, and may result in an LM highly relevant only to the specific training data. For example, one can fine-tune a large pre-trained LM on product reviews labeled with sentiment to generate positive and negative sentences, but the fine-tuned model will tend to generate sentences like those from product reviews which greatly limits its utility with out-of-domain prompts. Both these methods require training all the parameters of the model. Alternatively, recent research leverages a discriminator to re-weight output distributions [54] or to perturb latent representations in the token level such as in PPLM [55] without changing the pre-trained LM. However, raising target-relevant token probabilities may lead to less fluent sentences. In addition, updating gradients at the token level makes decoding expensive and slow.

Inspired by language codes which guide multilingual translation models to translate to the target language [56], we learn high-level attribute representation that can identify similarities and differences across different attributes. The main idea is to encode an *attribute* (e.g. positive, negative, business, military, etc.) with a pre-trained LM and learn an alignment function to transform the attribute representation. Examples of controlled generation based on high-level attribtues can be seen in Table 3.1.

In terms of more abstract features such as language representation, the goal is to easily adapt and transfer from one domain (or language), to another domain. Instead of an alignment function as a global indicator, we learn representations with vectors that capture cross-lingual similarities and differences across different dimensions. This information can guide a multilingual model regarding what and how much of the information in the model should be shared among specific languages. Different from previous research on generating language embeddings using prior knowledge such as word order [57, 58], using a parallel corpus [59, 60], and using language codes as an indicator to distinguish input and output words in a shared vocabulary into different languages [61, 32], our work focuses on generating and using language embeddings more effectively as soft-sharing [62] of parameters among various languages in a single model.

3.3 Related Work

3.3.1 Controlled Text Generation

To interpolate a controlling factor, concatenating the attribute to the input sequence is the most straightforward approach and has been commonly used in grounded generation [63, 64]. [53] proposes to pre-train a large conditional language model with available labels such as URLs for large LM control. This method can be effective in conditional modeling, but requires a substantial amount of resources for pre-training and is limited by the labels used during pre-training (e.g. 55 control codes in CTRL). Another approach is to concatenate the attribute representation to the hidden states using linear transformation [65, 66] or latent variables [67, 68]. These approaches require training from scratch or fine-tuning the entire pre-trained model to incorporate the external target attributes and model conditional probability [69, 70, 71]. In addition, they always require carefully designed Kullback-Leibler (KL)-Divergence and adversarial training to generate out-of training domain text with the desirable attribute only [72]. In comparison, our proposed method does not require fine-tuning the original LM so that we can make use of the high quality pre-trained LM while controlling the target attributes.

Instead of fine-tuning the whole model, [73] proposes to add residual adapters, which are task-specific parameters to transformer layers for each language understanding task. Different from adding adapters for each individual attribute [74, 75], our method only requires learning one attribute alignment function for all attributes to do controlled generation, and is more flexible at inference time without degrading quality such as diversity [76]. Recently, [77] proposes to use self-supervised learning with hand-crafted phrases (e.g. "is perfect" to represent positive sentiment), but suffers from high variance, low coherence and diversity in order to incorporate the target phrase. An alternative is to take a pre-trained unconditional LM and perturb the hidden states towards a target attribute in a plug and play manner [78]. PPLM proposes to train a classifier or bag-of-words to increase the likelihood of the target attribute in the hidden state for each token [55]. Similar to ours, their method does not require changing the pre-trained LM and they are able to control sentiment and various topics. However, ascending conditional probability

in the token level to shift the distribution towards target-related tokens can lead to degeneration [79] and is slow at inference time. The most similar work to ours is probably GeDi [80] which proposes to apply weighted decoding using class-conditional LMs with Bayes' Rule on each token to solve the slow inference problem. Concurrently, [81] introduces learning prefix rather than task instructions [9] and achieves better performances than adapter-based lightweight baselines. In contrast, our method learns an alignment function on hidden representations of the attribute so that tokens can do self-attention with the attribute without breaking the pre-trained self-attention in the LM. During generation, we can simply send the attribute as a signal for conditional generation. Our method is uniform for different attributes such as sentiment and topics, and is more efficient and flexible.

3.3.2 Language Representation

Feature-based language representations An intuitive method to represent language information is through explicit information such as known word order patterns [57, 82], part-of-speech tag sequences [83], and syntactic dependencies [84]. [58] propose sparse vectors using pre-defined language features such as known typological and geographical information for a given language. However, linguistic features may not be available for less studied languages. Our proposed approach assumes no prior knowledge about each language, deriving typological information from plain text alone. Once a vector for a target language is created, it contains many typological features of the target language, and can be used for transfer learning in downstream tasks.

Dense representation with parallel data Other previous work has also explored dense continuous representations of languages. One method is to append a language token to the beginning of a source sentence and train the language embeddings with a many-to-one neural machine translation model [85, 86]. Another method is to concatenate language embedding vectors to a character level language model [60, 87, 88]. These two methods require parallel translation data such as Bible and TED Talk. [89] derive typological information in the form of phylogenetic trees from translation of different languages into English and French using the European Parliament speech corpus [90], based on the

assumption that unique language properties are present in translations [91, 92]. [59] abstract the translated sentences from other languages to English with part-of-speech tags, function words, dependency relation tags, and constituent tags, and train the embedding vectors by concatenating a language representation with a symbol representation. In comparison, we generate our language embeddings using no parallel corpora or linguistic annotation, which is suitable for a wider variety of languages, including in situations where no parallel data or prior knowledge is available.

Language vectors without parallel data The approach that is closest to ours is XLM [32], which adds language embeddings to each byte pair embedding using Wikipedia data in various languages with a masked language modeling objective. However, similar to [61], the trained language embeddings only serve as an indicator to the encoder and decoder to identify input and output words in the vocabulary as belonging to different languages. In fact, in a follow up paper, XLM-R [93], language embeddings are removed from the model for better code-switching, which suggests that the learned language embeddings may not be optimal for cross-lingual tasks. In this paper, following the finding that structural similarity is critical in multilingual language models [94], we generate language embeddings from a denoising autoencoder objective and demonstrate that they can be effectively used in cross-lingual zero-shot learning.

3.4 Methodology

The general methodology we propose to learn high-level attribute representation is to learn some dense representations which can indicate to the model how to encode and decode corresponding texts. Specifically, as illustrated in Figure 3.1, we consider each attribute as some fixed length parameters, and train the parameters with tasks such as language modeling. Depending on the task and architecture, the attribute representations can be the only trainable parts while freezing the other parameters in the original model to utilize the original representation, or update the whole parameters as well. For example, in the Transformers architecture, we can initialize the representations at each layer and the tokens can attend to the learned representations as a guide towards the target direction. In

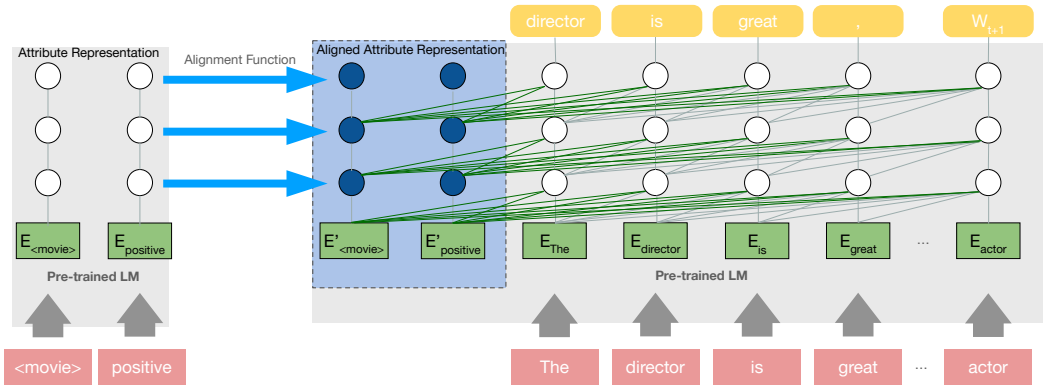

Figure 3.1. Attribute Alignment model architecture with corpus representation disentanglement. We train the alignment function (an MLP in our experiment shown as blue arrows) to transform attribute (e.g. `positive sentiment`) representation (encoder hidden states in the left grey box) to aligned attribute representation (blue shade box in the middle). The training objective is to generate attribute-related sentences in the training dataset by attending to aligned attribute representation (green lines) in addition to regular self-attention (grey lines).

the RNN architecture, we can choose to append a dense representation to the end of each token embedding. This process can be considered similar to learning soft prompts, which has the benefits of both easy to train, as well as saving storage. As we will show later, the learned attribute representations can be transferable across different architectures. In addition, if we do initialize the representation with some trained parameters and learn some additional transformation instead, we can easily swap in unseen attributes in a zero-shot setting, and combine many attributes together without limitations of restricting to training one specific representation per attribute.

By training the attribute representation this way, we can either use supervised parallel data, or train the model in a totally self-supervised manner. We will carefully describe the methodology for different types of high-level attributes and suggest how the representation should be trained efficiently in Section 3.5 and Section 3.6.

3.5 Language Representation for Cross-lingual Language Understanding

3.5.1 Generating Language Representations

We consider language representations and language embeddings. We first present the data used to generate language embeddings, then introduce our approach inspired by denoising autoencoders [95].

3.5.1.1 Data and preprocessing

To train our multilingual model, we use the CommonCrawl dataset from the CoNLL 2017 shared task [96] to obtain monolingual plain text in various languages. To represent words across different languages in a shared space, we use the unsupervised pretrained aligned word embeddings from MUSE [97]. We choose the 29 languages from the CoNLL 2017 monolingual text dataset for which MUSE pretrained embeddings are available.[1] A subset of 200K sentences are selected randomly for each language. The languages we use are: English, French, Romanian, Arabic, German, Russian, Bulgarian, Greek, Slovak, Catalan, Hebrew, Slovene, Croatian, Hungarian, Spanish, Czech, Indonesian, Swedish, Danish, Italian, Turkish, Dutch, Norwegian Bokmål, Ukrainian, Estonian, Polish, Vietnamese, Finnish, and Portuguese, which cover ten language genera.

We experiment with two types of word representations in training language embeddings. The most straightforward way is to use the pretrained MUSE embedding for each specific language (we refer to this setting as **Spe.**). We also experimented with mapping word embeddings from different languages into one language (English in our experiments because it is used as the pivot language in MUSE embeddings, **Eng.**) for three reasons. First, because MUSE is mainly trained by an orthogonal rotation matrix and the distances among words in each language are still maintained thereafter, language identities can potentially be revealed. The result is that the learned language embeddings reflect the features incorporated in the unsupervised word mapping methods instead of the intrinsic language features. Second, we hypothesize that mapping to a single language space requires the model to encode more information in language embeddings as their language

[1] `https://github.com/facebookresearch/MUSE`

identities instead of relying on their revealed ones. Finally, using shared word embeddings can reduce the vocabulary size for memory concerns by effectively reducing both the lookup table size and the output softmax dimension size.

For **Eng.** word embedding mapping, we align words from different languages to English embeddings using cross-domain similarity local scaling (CSLS, [97]). The vocabulary of our model is restricted to the words in the English MUSE embeddings, and all unknown words are replaced with a special unknown token. Although imperfect mapping from each language to English tokens may introduce noise and result in a coarse approximation of the original sentences, crucial syntactic and semantic information should still be present.

In our experiments, a language code is appended to each token according to the original language of the sentence. For instance, the German sentence "Er hat den roten Hund nicht gesehen" would be represented in our **Spe.** condition as

Er_de hat_de den_de roten_de Hund_de nicht_de gesehen_de

and in the **Eng.** condition as

he_de has_de the_de red_de dog_de not_de seen_de

Intuitively, the idea is to have the words themselves be the same across languages (either through the aligned MUSE embeddings or by direct mapping to English words), and let the additional language code provide to the model the information that would explain the structural differences observed across languages in the training data.

3.5.1.2 Denoising autoencoder

Given a multilingual plain text corpus with sentences in each language (and no parallel text), we first perturb each sentence to create a noisy version of the sentence where its words are randomly shuffled. The training objective is to recover the original sentences, which requires the model to learn how to order words in each language. We hypothesize that compared to language modeling, this will encourage the language embeddings to learn more structural information instead of relying on topics or word co-occurrence to generate meaningful training sentences. We implement our multilingual denoising autoencoder with

an LSTM-based [98] sequence-to-sequence model [34]. The input strings are perturbed sentences and the output strings are the original sentences.

After preprocessing the data, we concatenate a language embedding vector initialized from normal distribution as a language identity feature (the language code mentioned in Section 3.5.1.1) to each of the pretrained word embeddings. Since certain languages are more similar to, or more different from, each other, the model will learn how to reorder a sequence of words depending on the specific language. For example, reordering an Italian sentence should be more similar to reordering a Spanish sentence than it is to reordering a German sentence. Because the decoder captures the actual word order of the sentences in each target language, whereas the language codes in the encoder are meant to capture only language identity and no word order information, we use the extracted language embeddings from the decoder in our experiments.[2] Each word is represented with a pretrained 300-dimensional vector, and each language embedding is represented with a 50-dimensional vector[3]. The input token is thus a 350-dimensional vector from the concatenation.

3.5.2 Experiments

To examine the quality of the typological information captured by the language embeddings, we perform intrinsic and extrinsic evaluations. Our intrinsic evaluation consists of predicting linguistic typology and language features from the World Atlas of Language Structures (WALS, [99]). Our extrinsic evaluations are based on cross-lingual dependency parsing and cross-lingual natural language inference (XNLI, [100]) in a zero-shot learning setting, where a trained model makes predictions on a language not seen during training, but for which a language embedding has been learned from plain monolingual text. In contrast with previous research which applies learned typology to cluster similar languages and train machine translation tasks in clusters [86], we explore if we can apply the learned embeddings directly into downstream tasks. We compare three different sets of

[2]To confirm our assumption about the embeddings for the language codes in the encoder and the decoder, we also performed experiments using the encoder language embeddings. As expected, the results obtained with embeddings from the encoder were inferior in every case tested.

[3]We experimented with different dimensions for language embedding and did not observe performance difference.

embeddings based on our approach with three sets of embeddings from previous work:

Spe. lang_emb represents language embeddings from our proposed denoising autoencoder trained with language specific MUSE embeddings, using CommonCrawl text.

Eng. lang_emb represents language embeddings trained with English MUSE embeddings after mapping words from different languages to English, using CommonCrawl text.

Wiki lang_emb represents language embeddings trained with English MUSE embeddings using Wikipedia. We use the same data selection and preprocessing process as detailed in Section 3.5.1.1. We use these embeddings to show the impact of training data. In addition, we use these embeddings to compare with XLM embeddings trained with Wikipedia.

Malaviya represents language embeddings from [85], trained with a many-to-one machine translation model using Bible parallel data. It has 26 languages in common with our 29 languages except English, Hebrew, and Norwegian. We use these embeddings to represent previous methods of learning language representations from parallel data.[4]

XLM mono represents language embeddings trained with XLM model using the same monolingual data as Wiki lang_emb on 29 languages.

XLM parallel represents language embeddings trained with XLM using monolingual and parallel data from 15 XNLI languages. We extract the embeddings from the publicly available model.

3.5.2.1 Linguistic typology prediction

We first inspect the language embeddings qualitatively through principle component analysis (PCA) visualization. We also use spectral clustering to recover the language genus (language family subgroup) information from the embeddings. To compare the quality of the clusterings quantitatively, we calculate the adjusted Rand index [101] between the

[4]We do not evaluate the embeddings from [85] on parsing and XNLI because they do not include English embeddings, which are necessary for a direct comparison. In XNLI, in particular, there is only training data for English.

generated clusters and the actual language genera.

3.5.2.2 WALS feature prediction

We evaluate the language embeddings on predicting language features in WALS. Each WALS feature describes a characteristic of languages, such as the order of subject, object, and verb. We consider the features for which information is available for more than 50% of the languages we use and cast each feature prediction as a multi-class classification task. We then classify the features into the following categories.

- **Lexicon**: usage of specific words, e.g. whether the language has separate words for "hand" and "arm", etc.;
- **Syntax**: mostly related to the relative orders between various types of constituents, including order of subject, object and verb, adpositions and noun phrases, and also features related to syntactic constructions;
- **Partially Morphological (Part. Morph.)**: features that mainly concern syntax or semantics but either usually relate to morphology (such as inflectional morphemes), or have morphological information coded in the values of the features, e.g. gender systems, order of negative morphemes and verbs;
- **Non-learnable**: features that mainly concern morphology, phonology, or phonotactics, and are not learnable from reordering plain text.

The categories make it easier to evaluate what the language embeddings capture. We train linear classifiers to predict WALS results. For each feature, we hold out one language and train a classifier on the language embeddings of the rest of the languages to predict the corresponding feature values on the held-out language embedding, in a leave-one-out cross-validation scheme. We then average the accuracy of the features within each category to report the results. In addition to comparing different language embeddings, we also compare to two baselines: a **Random** baseline, and a **Majority** baseline (which predicts the most common value for each feature). We repeat this procedure 100 times while randomly permuting the orders of the input vectors to the classifiers to eliminate possible effects due to initial states and report the average and significant scores.

Compared to a recent shared task where the input is some features of a language (e.g. language family and various WALS features), with optionally pre-computed language embeddings to develop models to predict other features [102], we investigate if trained language embeddings alone can be used to predict WALS features. In addition, we showed that our language embeddings outperformed a frequency baseline among other baselines (see Section 3.5.3.2) compared to [102].

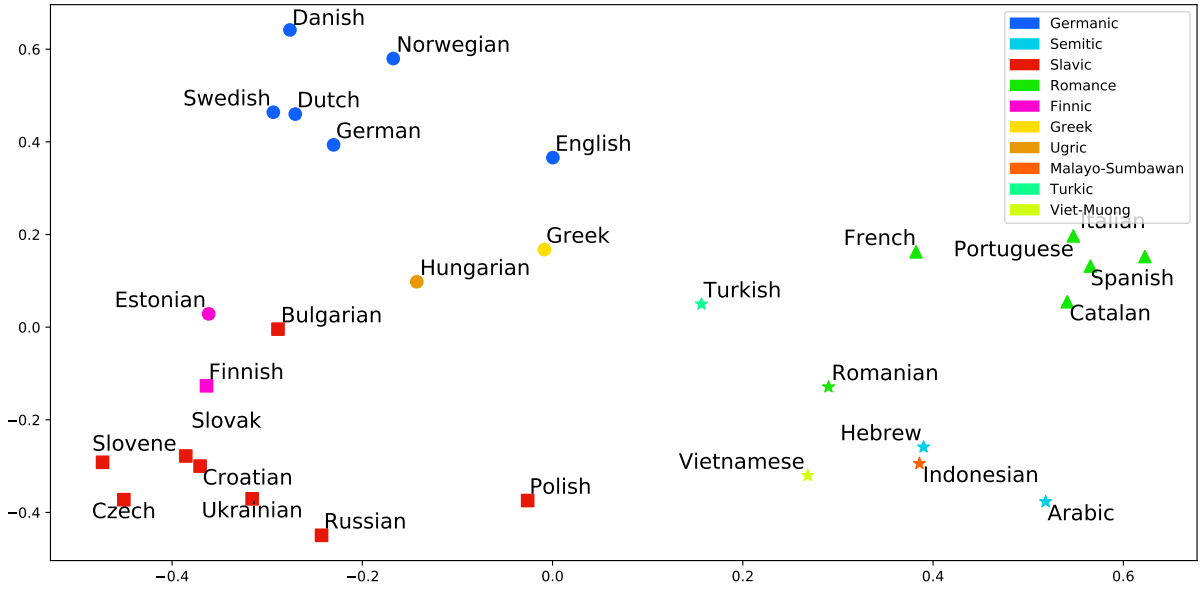

Figure 3.2. Two-dimensional PCA projection of the 50-dimensional language embeddings. Shapes represent automatically derived clusters, and colors represent language genera.

3.5.2.3 Cross-lingual dependency parsing

Since our language embeddings are trained using a word ordering task, we hypothesize that they capture syntactic information. To verify that meaningful syntactic information is captured in the language embeddings, we use a dependency parsing task where sentences for each target language are parsed with a model trained with treebanks from other languages, but no training data for the target language. This can be seen as a form of cross-lingual parsing or zero-shot parsing, where multiple source languages are used to train a model for a new target language. Without annotated training data for parsing a target language, the model is expected to leverage treebanks from other languages through language embeddings.

We use 16 languages from Universal Dependencies v2.6 [103], representing five distinct language genera (Table 3.3). We modified Yu Zhang's implementation[5] of biaffine dependency parser [104]. In specific, we freeze word embeddings, concatenate a 50-dimensional embedding (either the corresponding Eng. language embedding or a random embedding) to the embedding of each token, and not use part-of-speech information (since we are assuming no annotated data is available for the target language). The goal of this evaluation is not to obtain state-of-the-art attachment scores, but to find whether a model that uses our language embeddings produces higher attachment scores than a model that instead uses random embeddings of the same size[6]. While our embeddings should capture syntactic typology, random embeddings would simply indicate to the model the language for each sentence with no information about how languages are related.

3.5.2.4 XNLI

Natural language inference (NLI) is a language understanding task where the goal is to predict textual entailment between a premise and a hypothesis as a three-way classification: *neutral*, *contradiction*, and *entailment*. The XNLI dataset [100] translates English NLI validation and test data into 14 other languages. We evaluate on ten of the XNLI languages which we trained language embeddings with.

State-of-the-art models on XNLI are Transformers [105] pretrained on large corpora [106]. To evaluate if our learned language embeddings (from an LSTM model) can be plugged off-the-shelf into other architectures such as Transformer, we compare with two strong Transformer-based baselines, XLM ([32]. L = 12, H = 1024, 250M params) and XLM-R ([93]. XLM-R_{Base}: L = 12, H = 768, 270M params; XLM-R_{Large}: L = 24, H = 1024, 550M params). XLM adds language embeddings together with each word embedding and position embedding as the input embedding in training masked language modeling (MLM, with monolingual data) and/or a translation language modeling (TLM, with translation parallel data). In comparison, XLM-R removes language embedding and is pretrained with MLM on much more data. We train our model on the English MultiNLI [107] dataset,

[5] `https://github.com/yzhangcs/parser`

[6] Random embeddings are used to eliminate the effect of different dimensionality. In our preliminary experiments, we found that adding a random embedding performs better than not adding any embedding.

and directly evaluate the trained model on the other languages without language-specific fine-tuning, in a zero-shot cross-lingual setting. To select the best checkpoint for test set evaluation, we follow [93] by evaluating on the development set of all languages. In addition, we also experiment with a fully zero-shot transfer setting where we select the best checkpoint by evaluating on the English development set. We run the selected checkpoint on the test set of each language and report the accuracy scores. We use the public available XLM model pretrained on 15 XNLI languages with MLM and TLM objectives, and XLM-R pretrained on 100 languages. In order to add our learned language embeddings into XLM and XLM-R models, we normalize our embeddings to have the same variance as the XLM language embeddings, and we learn a simple linear projection layer to map our 50-dimension embeddings (which is frozen during training) to the hidden dimension of corresponding models. We report all results averaged over three random seeds.

3.5.3 Results and Analysis

We show results of our proposed language embeddings in comparison to the baselines and language vectors generated from previous work on linguistic typology, WALS, cross-lingual parsing, and XNLI. We report results with **Eng.** language embeddings.

3.5.3.1 Linguistic Typology

	Lexicon	Syntax	Part. Morph.	Non-learn.	Rand
n features	2	14	46	20	-
Random	0.56	0.61	0.52	0.52	-
Majority	0.64	0.75	0.69	0.68	-
Malaviya	0.66*	0.74	0.66	0.66	0.13
XLM mono	0.41	0.75	0.66	0.68	0.12
Spe.	0.64	0.78*	0.68	0.66	0.53
Eng.	0.85*	0.79*	**0.71***	0.66	**0.58**
Wiki	**0.87***	**0.81***	0.70*	**0.68**	0.51

Table 3.2. WALS prediction and linguistic typology clustering results on 26 in-common languages across 10 language genera. *indicates statistical significance ($p < 0.01$) over the Majority baseline.

Figure 3.2 shows a two-dimensional PCA projection of the learned language embeddings. Due to space limitations, we only show the projection of the language embeddings

using words mapped to English embeddings; using language-specific embeddings produces similar results. We can clearly see the clustering of Slavic languages on the lower left, Romance on the right, and Germanic on the upper left. Our dataset also contains two Finnic languages, which appear right above the Slavic languages, and two Semitic languages, which appear on the lower right. The other languages, Vietnamese, Indonesian, Turkish, and Greek, are from language groups underrepresented in our dataset, and appear either mixed with the Germanic languages (in the case of Hungarian, Turkish and Greek), or far on the lower right corner (Vietnamese, Indonesian). Romanian, a Romance language, appears miscategorized by our language embeddings. While it is close to the cluster of romance languages, it appears closer to the singleton languages in the dataset and to the two Semitic languages.

In addition to actual language relationships represented by color, we also present the result of spectral clustering with four categories through different shapes. Results illustrate that our language embeddings can capture similarities and dissimilarities among language families. In comparison, language embeddings generated by [85] do not capture clearly visible language relationships. Quantitatively, clusters from our learned language embeddings (**Eng.**) achieve a much higher Rand score (0.58) compared to previous language embeddings, as shown in Table 3.2 (last column). This indicates that our clusters closely align with true language families.

3.5.3.2 WALS predictions

Table 3.2 shows the prediction accuracy for WALS features, averaged within each category. Unlike the language representations generated by [59], which do not outperform the majority baseline without finetuning, our derived language embeddings perform significantly better than the baselines and previous methods in lexicon, syntax, and partially morphological categories. Note that even though the training objective of the denoising autoencoder is to recover a language-specific word order, the model does not use linguistic features such as grammatical relation labels or subject-verb-object order information. Instead, it derives typological information from text alone through the word reordering task. The language embeddings generated with words mapped to English embeddings

(**Wiki** and **Eng.**) generally produce more accurate predictions, with the models trained from Wikipedia producing slightly better results likely due to cleaner training data. Results from different settings show that we do not need clean data (e.g. Wiki) to generate language embeddings.

Language	Baseline	Language Emb.
Finnic		
Estonian	56.19	**61.68** (+5.49)
Finnish	59.59	**62.91** (+3.32)
Germanic		
Danish	63.31	**69.62** (+6.31)
English	**74.51**	74.08 (-0.43)
German	64.36	**65.67** (+1.31)
Norwegian	77.19	**78.20** (+1.01)
Slovene	67.92	67.91 (-0.01)
Romance		
Catalan	72.41	**80.76** (+8.35)
French	68.75	**79.37** (+10.62)
Spanish	74.42	**81.74** (+7.32)
Portuguese	71.11	**79.57** (+8.46)
Semitic		
Arabic	48.44	**52.51** (+4.07)
Hebrew	**41.87**	33.66 (-8.21)
Slavic		
Bulgarian	62.91	**67.00** (+1.09)
Czech	65.62	**66.98** (+1.36)
Russian	62.10	**66.45** (+4.35)
Average	64.61	**68.01** (+3.40)

Table 3.3. Zero-shot parsing results (UAS), where each of 16 languages are parsed using annotated language from the other 15 languages. In the *Language Emb.* column, results were obtained by concatenating the language embedding to each token's MUSE embedding. In the *Baseline* column, results were obtained using a random embedding instead. Boldface indicates a statistically significant difference ($p < 0.05$).

3.5.3.3 Cross-lingual dependency parsing

The cross-lingual dependency parsing results in Table 3.3 indicate that our language embeddings are in fact effective in allowing a parsing model to leverage information from different languages to parse a new language. Substantial accuracy improvements were observed for 13 of the 16 languages used in the experiment, while accuracy degradation was observed for two languages. Notably, there were large improvements for each of the four Romance languages used (ranging from 7.32 to 10.62 absolute points), and a steep

	fr	es	de	el	bg	ru	tr	ar	vi	avg.
Selected with English development set										
XLM	77.3	77.9	75.9	74.3	75.3	73.8	70.4	70.9	73.2	74.3
XLM + lang_emb	**78.3**	**79.0**	**76.5**	**75.6**	**76.6**	**74.8**	**71.3**	**72.3**	**74.4**	**75.4**
Selected with averaged development set										
XLM	77.4	78.2	76.1	75.4	76.3	74.4	70.3	71.7	73.5	74.8
XLM + lang_emb	**78.5**	**79.0**	**76.7**	**75.9**	**76.8**	**75.3**	**71.5**	**72.4**	**74.8**	**75.7**
XLM-R_{Base}	77.9	78.7	76.9	76.0	77.9	75.9	72.4	72.2	74.8	75.9
XLM-R_{Base} + lang_emb	**78.8**	**79.4**	**77.4**	**76.2**	**78.2**	**76.1**	**73.2**	**72.6**	**75.4**	**76.4**
XLM-R_{Large}	83.6	84.6	83.0	82.4	83.3	80.3	79.1	79.0	80.0	81.7
XLM-R_{Large} + lang_emb	**83.9**	**84.8**	**83.7**	**82.8**	**84.2**	**81.1**	**80.3**	**79.4**	**80.3**	**82.3**

Table 3.4. Results on XNLI test set with zero-shot prediction. The results show that adding language embeddings outperforms the baselines in all settings.

drop in accuracy for Hebrew (-8.21). Although a sizeable improvement was observed for the only other language from the same genus in our experiment (Arabic, with a 4.07 improvement), accuracy for the two Semitic languages was far lower than the accuracy for the other genera. This is likely due to the over-representation of Indo-European languages in our dataset, and the lower quality of the MUSE word alignments for these languages.

While our accuracy results are well below current results obtained with supervised methods (i.e. using training data for each target language), the average accuracy improvement of 3.4 over the baseline, which uses the exact same parsing setup but without language embeddings, shows that our language embeddings encode actionable syntactic information, corroborating our results using WALS.

3.5.3.4 XNLI prediction

The XNLI results in Table 3.4 indicate that our language embeddings, which capture relationships between each test language and the training language (English), are also effective in tasks involving higher-level semantic information. We observe consistent per-

formance gains over very strong baselines in all settings and models for each language. Specifically, in the fully zero-shot setting where we select the best model based on the English development data, adding our learned language embeddings increases 1.1 absolute points on average for XLM. The same trend holds for XLM-R results, not shown due to space limits. On the other hand, if we select the best model on the averaged development set following [93], we observe averaged performance gain of 0.9, 0.5, and 0.6 absolute points for XLM, XLM-R_{Base}, and XLM-R_{Large}, respectively. We conjecture that the lower improvement on XLM-R models compared to XLM is due to that XLM-R was pretrained without language embeddings. When we add our language embeddings to the original word and positional embeddings, the distribution of the overall input embedding such as variance is changed. Hence, the language embeddings can be considered as noise at the beginning, making it hard to learn and incorporate additional information. However, the improvement is consistent over all strong baselines, suggesting that our language embeddings, which are not optimized towards any specific task, can be leveraged off-the-shelf in large pretrained models and achieve better zero-shot transfer ability in downstream tasks.

3.5.3.5 Discussion

Our results in each of the intrinsic and extrinsic evaluation settings demonstrate that our denoising autoencoder objective, which has been shown to be effective in various language model pre-training tasks [35, 108], is effective for learning language embeddings that capture typological information and can be used to improve cross-lingual inference. Even though reconstructing the original sentence from a randomly ordered string is the direct training objective, our evaluation of the resulting embeddings is not based simply on word order.

The grammar of a language is of course an important factor in determining the order of words in a sentence in that language, although it is not the only factor. The syntax area features in our WALS evaluation, which are largely related to relative orders of constituents and syntactic constructions and therefore clearly relevant to our training objective, confirm that part of what our embeddings capture is in fact related to word ordering. However, our results on the lexicon and morphology areas indicate that

language-specific information capture in our embeddings goes beyond ordering information. Although it may seem that the model only has access to information about word ordering during training, text in the various languages also provides information about word usage, co-occurrence, and to some extent even inflection through the word embeddings. As a result, language embeddings trained with our approach capture interpretable and useful typological information beyond word order. Because language embeddings are the only signal to the model indicating what each of the languages that are mixed within the training data reads like, we conjecture that our denoising autoencoder objective encourages the embeddings to encode language-specific information necessary to distinguish each language from the others.

3.6 Feature Representation for Controlled Language Generation

3.6.1 Learning representation

Unconditional language models are trained to optimize the probability of $p(x_i|x_{0:i-1})$ where x_i is the next token and $x_{0:i-1}$ are already generated tokens. For controlled generation, we need to model the conditional distribution $p(x_i|x_{0:i-1}, \mathbf{a})$ where $\mathbf{a}$ is the attribute for the model to condition on. To make use of large LMs trained on unlabeled data, we need to infuse the attribute $\mathbf{a}$ into the pre-trained unconditional distribution $p(x_i|x_{0:i-1})$. We introduce `Attribute Alignment` to this end. Different from fine-tuning the whole LM, our alignment function is the only trainable component while the pre-trained LM parameters are frozen.

3.6.1.1 Attribute representation with alignment function (A)

The high-level idea is to append the attribute token to the beginning of a prompt as a signal so that each token in the sentence can attend to the attribute token. However, this may break the originally learned sequential dependencies because now the sentence starts with an attribute token followed by a regular sentence, different from the data used for large LM pre-training.

Instead, `Attribute Alignment` first gets the hidden states of the attribute by running

the pre-trained LM on **a**. Then we align the hidden states using our alignment function ($\mathcal{F}$), implemented as a multi-layer perceptron (MLP) with non-linear connections in this paper, to get aligned attribute representation. Specifically, in the Transformer architecture [105] where hidden states are represented as key-value pairs, the key (K) and value (V) pair after attribute representation alignment is represented by

$$K'_{:t}, V'_{:t} = [\mathcal{F}(K_{\mathbf{a}}); K_{:t}], [\mathcal{F}(V_{\mathbf{a}}); V_{:t}] \tag{3.1}$$

$K_{\mathbf{a}}$, $V_{\mathbf{a}}$ are from $LM(x_{\mathbf{a}})$ and $K_{:t}$, $V_{:t}$ are from $LM(x_{:t})$ where $x_{\mathbf{a}}$ is the attribute phrase, and $x_{:t}$ are the tokens in the generated sentence up to timestep t. Then we can calculate attention and output in the original Transformer model.

During training, we freeze the pre-trained LM and compute the language modeling loss on datasets with the attribute **a** to train the alignment function $\mathcal{F}$. The loss function is thus

$$\mathcal{L}_A = -\sum_{t=0}^{l} \log p(x_t|\mathbf{a}, x_{:t}) \tag{3.2}$$

and we only update the parameters of the alignment function using the gradients. Fig.3.1 illustrates the model architecture. At inference time, all tokens starting from the prompt attend to the target attribute representation transformed by the trained alignment function in addition to the standard self-attention to generate the next token. Intuitively, this can be considered as a conditional LM because all tokens now can attend to the aligned attribute representation.

3.6.1.2 Disentangle irrelevant attributes

The learned alignment function bridges the attribute representation to pre-trained LMs. However, we do not disentangle different features in the training data. For instance, if we train the alignment function on a movie review dataset for sentiment control, then $\mathcal{F}$ encodes both sentiment and movie review style after aligning the sentiment attribute representation. Thus, the target attribute representation may be diluted. To solve this problem, we propose three disentanglement methods.

Attribute representation with corpus representation disentanglement (AC) We propose to add a corpus domain representation **d** along with the attribute represen-

tation **a** during training. For a training corpus (such as movie reviews) with multiple attributes (such as positive and negative sentiment), **d** is used in all the training data while **a** is only used in a subset of the training data labeled with the target attribute. Similar to [109], this can encourage the model to encode target attribute and other features separately into different representations. Specifically, the key-value pairs can be represented as

$$K''_{:t}, V''_{:t} = [\mathcal{F}(K_{\mathbf{a}}); \mathcal{F}_{\mathbf{d}}(K_{\mathbf{d}}); K_{:t}], [\mathcal{F}(V_{\mathbf{a}}); \mathcal{F}_{\mathbf{d}}(V_{\mathbf{d}}); V_{:t}] \tag{3.3}$$

where $\mathcal{F}_{\mathbf{d}}$ is a separate alignment function for corpus domain representation, and $K_{\mathbf{d}}$, $V_{\mathbf{d}}$ are from the LM encoding of corpus domain names. Compared to attributes, corpus domain names might be more abstract so we use special tokens for **d** (such as `<movie review>`) and the original texts for attributes (such as `athlete`). At inference time, we want to generate coherent sentences given any (including out-of-domain) prompts. Therefore, we ignore the corpus representation while having tokens attend to the attribute representation in addition to normal self-attention as in Equation 3.1 [7].

KL disentanglement (ACK) We also experiment with adding KL-Divergence on top of AC to ensure that the LM does not diverge too much from the original distribution when an attribute signal is added following [55]. The disadvantage of this method, however, is that KL-Divergence may also prevent the alignment function from learning useful updates to attribute representation.

Bayes disentanglement (ACB) To further disentangle different features, we use Bayes' Rule to split domain-relevant distribution from attribute-relevant distribution. Derived from Bayes' Theorem, we have

$$p(x|\mathbf{a}) \sim \frac{p(x|\mathbf{a},\mathbf{d})}{p(x|\mathbf{d})} \cdot \frac{p(x,\mathbf{a})}{p(\mathbf{a}|x,\mathbf{d})} \tag{3.4}$$

$p(x|\mathbf{a},\mathbf{d})$ is the probability distribution of the generated sentence conditioning on both the attribute and the corpus domain, while $p(x|\mathbf{d})$ is the probability distribution of the generated sentence conditioning on the corpus domain only. During training, we assume

[7] In other words, if corpus representation is considered, generating movie reviews or wikipedia-type sentences for any prompt will greatly limit its utility

that different attributes in a corpus (e.g. different sentiments in movie reviews) are close to a uniform distribution. Hence, we consider $p(a|x,d)$ as a constant for a given sentence x from the corpus d. Likewise, we consider $p(x,a)$ as a probability distribution from the frozen pre-trained LM with roughly comparable attribute distribution on any sentence to approximate $p(a|x)$, similar to [110]. Therefore, we approximate this equation by eliminating the rest where the elimination does not directly impact a specific training sentence for the target conditional distribution. We can approximate the desired conditional probability in the log space as

$$\log p(x|\mathbf{a}) \sim \log p(x|\mathbf{a},\mathbf{d}) - \log p(x|\mathbf{d}) \tag{3.5}$$

During training, we train the attribute and domain alignment functions $(\mathcal{F}, \mathcal{F}_{\mathbf{d}})$ by running the LM conditioned on both attribute and domain $(p(x|\mathbf{a},\mathbf{d}))$, and on domain only $(p(x|\mathbf{a}))$. In specific, the loss function is

$$\mathcal{L}_{ACB} = -\sum_{t=0}^{l} \log p(x_t|\mathbf{a},\mathbf{d},x_{:t}) + \sum_{t=0}^{l} \log p(x_t|\mathbf{d},x_{:t}) \tag{3.6}$$

Similar to other proposed methods, the loss is used to update $\mathcal{F}$ and $\mathcal{F}_{\mathbf{d}}$. At inference time, suggested by [110], we use a hyper-parameter λ to balance the two distributions. Therefore, the distribution we sample tokens from is

$$\log p(x|\mathbf{a}) \sim \log p(x|\mathbf{a},\mathbf{d}) - \lambda \log p(x|\mathbf{d}) \tag{3.7}$$

3.6.1.3 Multi-attribute Control and Zero-shot Inference

We can simply concatenate aligned attribute representations to control multiple attributes at the same time. In addition, as we learn the alignment function on the attribute hidden representation from word embeddings instead of learning the attribute representation directly [75], we can switch in any attribute token at inference time. Therefore, we can choose attributes not seen in the training corpus and generate text conditioned on a new topic as a zero-shot setting.

3.6.2 Experiments

We evaluate our proposed methods **A**: using attribute representation only; **AC**: Model A with corpus representation for disentanglement; **ACK**: AC with KL disentanglement; and

lastly **ACB**: AC with Bayes disentanglement. We evaluate these models on sentiment control for thorough comparisons. We use nucleus sampling [79] for all the methods at inference time.

3.6.2.1 Sentiment control

Data. We use the Stanford Sentiment Treebank (SST, [111]) as our training data. We choose the sentences with positive and negative sentiment to train our alignment function. We select the same 15 prompts such as "Once upon a time" that were used in prior work, which were originally randomly selected [55].

Baselines. We compare with five baselines. **GPT2** generates unconditioned sentences given the prompts from pre-trained GPT2-medium. The generated sentences are coherent and consistent, but may not capture the target attribute. Its fluency, diversity, and how much the results look like a particular training corpus serve as an upper bound. **GPT2-concat** appends the sentiment token (i.e., `positive`, `negative`) before the prompt. It shares the same motivation as our model (see Section 3.6.1.1). **GPT2-finetune** is GPT2 fine-tuned with all the model parameters on the same SST dataset by appending an attribute token to the beginning of a sentence. Its sentiment control score is an upper bound. **PPLM** perturbs pre-trained LMs to incorporate attributes without fine-tuning the LM parameters. Similar to ours, the recent state-of-the-art **GeDi** incorporates target attributes by weighted decoding on the token-level and uses Bayes' Rule on all control codes (rather than domain) to remove unwanted attributes. It serves as a strong baseline.

3.6.2.2 Topic control

Data. For topic control, we use AG News dataset [112] with four topic attributes ("World", "Sports", "Business", "Sci/Tech") and DBpedia [112] with 14 topic attributes such as "natural place" as our training data. We use the same 20 prompts from [55]. AG News dataset collects news articles whereas DBpedia dataset collects entity definitions from Wikipedia.

Baselines.

PPLM uses different methods for topic control (pre-defined bag of words). For fair comparison, we only compare with **GPT2**, **GPT2-finetune**, and **GeDi** training on the same data. We choose the best preforming models from sentiment control for topic

control experiments (**AC**, **ACB**), while having ablation study among proposed models on sentiment control.

3.6.2.3 Evaluation

We evaluate our proposed methods and baselines on sentiment and topic control. Following [55], we sample ten sentences in a batch and select the most attribute-relevant one over three runs for human evaluation for each prompt in each target attribute. For automatic evaluation, we compare the average performance on all the 30 (3×10) conditionally generated results to test the average performance and stability against variances.

Automatic evaluation We evaluate the conditional generation results on fluency, diversity, attribute relevance, and training data corpus resemblance.

Fluency is measured by GPT2-large, a pre-trained external LM, different from the LM we conduct our experiments with (GPT2-medium). We get the average perplexity of the generated sentences (including the prepended prompt). The perplexity score also indicates how much the generated examples diverge from the pre-trained LM.

Diversity is measured by distinct uni-, bi-, and tri-gram ratios as Dist-1, Dist-2, and Dist-3 [110] averaged over all generated sentences.

Attribute relevance measures how well the generated examples condition on the target attributes. We train classifiers to predict the probability that a given sentence has the target attribute. For sentiment control, we train an external sentiment classifier using IMDB movie review dataset [113] with a BERT [7] classifier. The classifier achieves an accuracy of 88.51% on the IMDB test set. We also experiment with an internal sentiment classifier trained with SST development set, and we observe that the prediction on the generated texts is similar to that with the external classifier.

For topic control, we train multi-class classifiers with BERT using 80% of the development sets of AG News and DBpedia datasets. The classifiers achieve an accuracy of 89.71% and 99.25% on the rest of the two development sets, respectively. Because other datasets do not share the same topics, we cannot train external classifiers.

Training data corpus resemblance is used to evaluate if the proposed methods generate sentences that contain undesirable features such as style from the training corpus. For

Model	Attribute			Quality				Data
	Sentiment% ↑	Positive% ↑	Negative% ↑	PPL↓	Dist-1 ↑	Dist-2 ↑	Dist-3 ↑	Corpus resemblance % ↓
Baselines								
GPT2	49.24	77.15	21.33	37.78	**0.49**	**0.85**	**0.91**	**18.31**
GPT2-concat	52.24	65.42	39.06	57.5	0.49	0.84	0.89	18.87
PPLM	57.03	81.58	32.47	54.03	0.44	0.79	0.88	26.12
Attribute Alignment								
A	52.61	77.35	27.86	40.19	0.45	0.82	0.90	59.13
AC	68.92	80.22	57.61	48.78	0.47	0.84	**0.91**	62.13
ACK	64.89	76.25	53.53	52.66	0.48	0.84	**0.91**	62.8
ACB	64.49	85.35	43.64	36.62	0.48	**0.85**	**0.91**	24.05
Attribute Alignment with strong polarized training data								
AC-S	67.04	81.62	54.45	38.46	0.45	0.80	0.88	63.21
ACB-S	58.85	80.88	36.82	**33.33**	0.46	0.83	0.89	28.12
Language model fine-tuning								
GPT2-finetune	**78.78**	81.92	75.63	55.60	0.37	0.66	0.75	92.24

Table 3.5. Results on sentiment control. Our proposed model with Bayes disentanglement (ACB) achieves good performance on sentiment controlling while maintaining high quality language generation. Note that even though GPT2-finetune achieves the best sentiment controlling score by training the whole LM, it suffers in generation quality and the generated sentences read like movie reviews measured by corpus resemblance.

instance, because our proposed method trains with a movie review dataset, the generated examples may tend to be semantically similar to movie reviews. Similar to attribute relevance, we train a BERT classifier by randomly selecting 2,000 training examples and 500 development examples from each of SST, DBpedia, and AG News, and the trained classifier achieves an accuracy of 99.3%. We report the probability that a generated sentence is from its controlling attribute training corpus as the corpus resemblance score.

Human evaluation We evaluate the generated sentences on attribute relevance, language quality, and training data corpus resemblance. All the metrics are on 1-5 Likert scale. **Attribute relevance** and **Corpus resemblance** are similar to the automatic metrics, measuring the degree to which the generated sentences are relevant to the target attributes, and how much the generated sentences read like from their corresponding training corpus, respectively. Since one can easily increase attribute relevance score

by sampling target-related tokens more frequently regardless of coherence and the context, **Language quality** measures if the generated sentences are coherent, in addition to fluency. Since GeDi outperforms previous strong baselines including PPLM from both automatic and human evaluation [80], we only do human evaluation comparing our best performing model (ACB) with GeDi.

3.6.3 Results and Analysis

We show controlled examples in Table 3.1 and analyze sentiment and topic control results as follows.

3.6.3.1 Sentiment control

Comparison with baselines. Table 3.5 shows results on sentiment control. Compared to the pre-trained LM (GPT2, 49.24%), all our proposed methods achieve better sentiment controlling scores with a large margin and get similar distinct scores. This shows that our proposed method is effective in sentiment control.

Even though GPT2-finetune achieves the highest sentiment score (78.78%), it gets higher perplexity, lower distinct scores, and very high corpus resemblance (92.24%). This implies that we can fine-tune a pre-trained LM to condition on the target attribute but suffer from the cost of being restricted to generating sentences resembling the training data.

All our methods outperform PPLM and GeDi with better sentiment control and diversity while having higher language quality. For qualitative comparisons between our proposed method and PPLM, we use the IMDB classifier to rank the most negative sentence generated from 30 examples for each prompt. Compared to our models, PPLM suffers from repetition and degeneration problems suggested by both distinct scores and qualitative analysis from the generated examples. Similarly, even though GeDi can successfully generate sentiment relevant sentences with prompts similar to the training data (such as "The book" for book reviews, [80]), it does not generate coherent examples with target sentiment (2.18 from human annotation) on a more diverse set of prompts. In contrast, using the aligned attribute representation as a control signal to guide the text generation leads to higher sentiment controlling probabilities while keeping the original

Topic source	Model	Attribute	Quality				Data
		On topic prob. % ↑	Perplexity↓	Dist-1 ↑	Dist-2 ↑	Dist-3 ↑	Corpus resemblance % ↓
AG News	GPT2	25.43	38.00	**0.49**	**0.84**	**0.90**	**84.08**
	AC	63.38	32.37	0.47	0.83	**0.90**	92.47
	ACB	**64.80**	**31.22**	0.46	0.83	**0.90**	90.66
DBpedia	GPT2	6.63	**37.40**	0.49	**0.84**	**0.90**	**1.01**
	AC	**32.98**	60.22	**0.50**	**0.84**	**0.90**	7.33
	ACB	32.18	49.85	0.49	0.83	**0.90**	7.46

Table 3.6. Topic control results with topics from AG News and DBpedia. Our proposed methods outperform the baselines by a large margin while having similar perplexity and diversity compared to the pre-trained language model (GPT2).

quality.

Comparison among proposed methods. The worse performance of having attribute representation only (52.61%) indicates that the entangled attributes dilute the conditional distribution and result in texts using similar vocabularies suggested by low diversity scores. In comparison, adding a corpus representation to disentangle target attributes leads to the best performance on sentiment probability prediction. Further disentanglement by adding KL-Divergence and separating corpus distribution with Bayes' theorem helps to reach lower perplexity and higher distinct scores as expected, but it hurts the attribute controlling performances. This may be caused by that the attribute and corpus representations in fact still mingle with each other so that when we remove the corpus distribution, we also remove some of the target attribute distribution. We also note that without Bayes disentanglement, all the other proposed methods reach much higher training corpus resemblance score (e.g. 62.13% with AC) but still much lower than that from fine-tuning (92.24%). This may be partially explained by that sentences with a strong sentiment are more similar to movie reviews than others from the training corpus resemblance classifier. Combining all the metrics, it shows that there is trade-off between sentiment control and generation quality. However, we can still control the sentiment better without the cost of perplexity, diversity, and style convergence than the strong baselines.

Adversarial prompts results. Following [55], we also experiment with generating a

sentence to an opposing sentiment from a highly polarized prompt. For example, the goal is to generate a positive sentence with the negative prompt "The food is awful". Using the external classifier to select the generated examples with the most likely target sentiment, we can obtain sentences such as "The food is awful but the service is amazing!" which is coherent compared to methods like PPLM and GeDi perturbing on the token level. Despite the prompts being very polarized, our method can still lead the text generation to the target sentiment without compromising fluency and diversity. More importantly, although we train our alignment function in the movie review domain, our generated sentences are not biased towards the domain.

Attribute data influence results. To evaluate how much attribute relevance in train-

Model	Attribute			Quality				Data
	Sentiment% ↑	Positive% ↑	Negative% ↑	PPL↓	Dist-1 ↑	Dist-2 ↑	Dist-3 ↑	Corpus resemblance % ↓
AC-S	67.04	81.62	54.45	38.46	0.45	0.80	0.88	63.21
ACB-S	58.85	80.88	36.82	**33.33**	0.46	0.83	0.89	28.12

Table 3.7. Results on sentiment control comparing strong polarized training data.

ing data influences controlling effect, we experiment with training on strong polarized examples labeled as "very positive" and "very negative" from SST. We denote the corresponding models as **AC-S**: AC with strong polarized training data; and **ACB-S**: AC-S with Bayes disentanglement. Table 3.7 shows that training with strong polarized data achieves similar controlling ability but suffers from lower diversity. This suggests that our proposed method is not sensitive to the attribute quality in the training data, showing the potential to use less strictly annotated data for controlling more diverse attributes.

3.6.3.2 Topic control

Comparison among different methods. We present our results on topic control in Table 3.6. Similar to sentiment control, we observe that our proposed methods significantly outperform the baseline in target topic controlling while holding similar perplexity and distinct scores. Even though the topic relevance score is lower than GeDi from automatic evaluation, ACB performs similarly measured by human annotation in terms of both relevance and language quality, while being much more diverse. In addition, us-

ing Bayes' disentanglement results in lower perplexity. However, compared to sentiment control, further disentanglement derives controlling effect on par with the simple disentanglement (+1.42% and −0.80% relative change for AG News and DBpedia) and generates comparable distinct scores. This indicates that topic attribute representations may be less entangled with other features such as style from the training corpus compared to that for sentiment representation.

Comparison between training dataset. To compare the results between topics from AP News and DBpedia, the perplexity is higher than the baseline and the relative corpus resemblance score is also high for DBpedia. We conjecture that this is caused by that topics such as "educational institution" may be difficult to associate with prompts such as "Emphasised are" in the pre-trained LM. When we control the model to generate sentences with the corresponding attributes, the generation diverges from the pre-trained LM more. However, distinct n-grams are not sacrificed.

3.6.3.3 Comparison to GeDi

From both sentiment control and topic control, we can see that our propose method is on par or better than GeDi in terms of attribute relevance and language quality, while being much more diverse (more than 10% averaged absolute points on distinct scores). Qualitatively, because GeDi applies weighted decoding on the token level similar to PPLM, we observe that it indeed boosts attribute-relevant token distribution which may lead to incoherent sentences (such as repeating the same phrase). For instance, regardless of the prompt, country and names (e.g. "Palestinian") are frequently sampled for the attribute "world". This can be further justified by their lower diversity score compared to the baselines. In addition, since GeDi utilized Bayes' Rule on all attribute codes (in comparison to ours on domains), it can also explain the lower performance on sentiment control where attributes are less decoupled.

3.6.3.4 Multi-attribute control and zero-shot analysis

In Table 3.1, we show examples with controlling multiple attribute (e.g. "world + science technology"). In addition, topics such as "military" are not in the topic control training corpus so that they are considered as zero-shot attributes. Our trained alignment function

can map unseen attribute representation to the target representation to generate fluent and on-topic sentences. However, this zero-shot ability largely depends on the unseen attribute and the provided prompt. Following previous research [53] where there may not be good evaluation metrics for the much harder multi-attribute and zero-shot inference task, we only show generated examples here with limited human annotation results showing better controlling and language quality compared to previous work [80]. We conjecture that our better performance is due to our more flexible alignment structure. In comparison, it is more complicated to compute the contrastive generation decoding method using Bayes rule suggested by [80] with more control codes without compromising the marginal distribution.

3.7 Summary

In this chapter, we propose methods to learn high-level attribute representations. We show examples of learning one representation per attribute, or learning alignment functions to convert original attribute representations (embeddings) to be generalizable. We showed that on sentiment and topic attribute representations, we can successfully convert an original language model into a controllable model through attribute alignment, while maintaining the language generation quality. Moreover, we demonstrated that we can efficiently control a trained model for transfer learning by leveraging similarities in more abstract language representations. Our method suggests that we do not require any supervised data, or only need a small parallel corpus, which is a promising direction to adapt a trained model to data-scarce scenarios such as cross-lingual settings where there may not be labeled data for a specific language, or we need to control a model with multiple unseen attributes.

Chapter 4
Low-level Attribute Representation

4.1 Overview

Not all attributes can be easily defined such as a positive and negative sentiment, or different languages, where each attribute is fundamentally different from others. In this chapter, we use an example of ontology attribute to illustrate how we can learn the attribute implicitly and generate their corresponding representations.

This chapter is based on our work [114] that was published at the NAACL 2022 conference, which I lead as the main author. Mingqiu Wang, Yuan Cao, Izhak Shafran, Laurent El Shafey, and Hagen Soltau (all from Google) serve in an advisory capacity.

4.2 Schema Induction

Defining task-specific schemas, including intents and arguments, is the first step of building a task-oriented dialog (TOD) system. In real-world applications such as call centers, we may have abundant conversation logs from real users and system assistants without annotation. To build an effective system, experts need to study thousands of conversations, find relevant phrases, manually group phrases into concepts, and iteratively build the schema to cover use cases. The schema is then used to annotate belief states and train models. This process is labor-intensive, error-prone, expensive, and slow [115, 116, 117, 118]. As a prerequisite, it hinders quick deployment for new domains and tasks. We therefore are interested in developing automatic schema induction methods

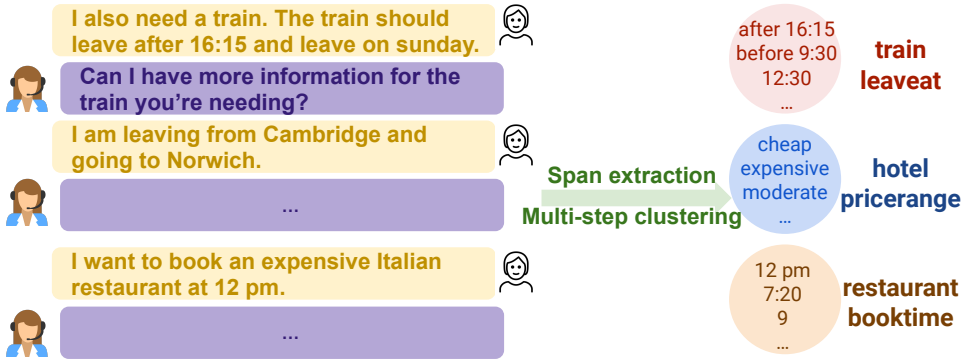

Figure 4.1. Overview of slot schema induction from raw conversations. We use a bottom-up representation level distance function derived from pre-trained LMs (combined with PCFG structure) to extract informative candidate phrases such as "after 16:15" and "expensive". The spans are subsequently clustered through multiple stages to form coarse to fine categories. The ground truth mapping is shown on the right (such as "train leaveat").

in this work to create the ontology[1] from conversations for TOD tasks.

Most existing approaches for slot schema induction rely on syntactic or semantic models trained with labeled data [119, 120, 117]. Our proposed method, on the other hand, is completely unsupervised without requiring generic parsers and heuristics, and hence portable to new tasks and domains seamlessly, overcoming the limitations of previous research. Analogous to human experts, our procedure is divided into two general steps: relevant span extraction, and slot categorization. Fig. 4.1 provides an overview of our approach. We introduce a bottom-up span extraction method leveraging a pre-trained language model (LM) and regularized by unsupervised probabilistic context-free grammar (PCFG) structure. We also propose a multi-step auto-tuned clustering method to group the extracted spans into fine-grained slot types with hierarchy.

We demonstrate that our unsupervised induced slot schema is well-aligned with expert-designed reference schema on the public MultiWoZ [121] and SGD [122] datasets. We further evaluate the induced schema on dialog state tracking (DST) and response generation to indicate usefulness and demonstrate performance gains over strong supervised baselines. Meanwhile, our method is applicable to more realistic scenarios with complicated schemas.

[1]We use "schema" and "ontology" interchangeably in this paper. Following previous work in literature, we focus on schema induction for slots, which is more challenging than domains and intents.

4.3 Related Work

Schema induction for dialog Motivated by the practical advantages of unsupervised schema induction, [123, 124] propose to induce spoken dialog grammar based on n-grams to generate fragments. Different from studying semantic grammars, [119, 125, 126, 127, 120] propose to utilize annotated FrameNet [128] to label semantic frames for raw utterances [129]. The frames are designed on generic semantic context, which contains frames that are related to the target domain (such as "expensiveness") and irrelevant (such as "capability"), while other relevant slots such as “internet” cannot be extracted because they do not have corresponding defined frames. This line of work focuses on ranking extracted frame clusters and then manually maps the top-ranked induced slots to reference slots. Instead of FrameNet, [130] extract features such as noun phrases (NPs) using part-of-speech (POS) tags and frequent words and aggregate them via a hierarchical clustering method, but only about 70% target slots can be induced. In addition to the unsatisfactory induction results due to candidate slot extraction, most of the previous works are only applicable to a single domain such as restaurant booking with a small amount of data, and require manual tuning to find spans and generate results. These methods are not easily adaptable to unseen tasks and services.

The most comparable work to ours is probably [117], which is not bounded by an existing set of candidate values so that potentially all slots can be captured. They propose to mix POS tags, named entities, and coreferences with a set of rules to find slot candidates while filtering irrelevant spans using manually updated filtering lists. In comparison, our method does not require any supervised tool and can be easily adapted to new domains and tasks with self-supervised learning. In addition to flexibility, despite our simple and more stable clustering process compared to their variational embedding generative approach [131], our method achieves better performance on slot schema induction and our induced schema is more useful for downstream tasks.

Span extraction Previous works in span extraction consider all combination of tokens as candidates [132]. Alternatively, keyphrase extraction research [133, 134] mostly depends on corpus statistics (such as frequency), similarity between phrase and document

embeddings, or POS tags [135, 136], and formulates the task as a ranking problem. Although these methods can find meaningful phrases, they may result in a low recall for TOD settings. For instance, the contextual semantics of a span (such as time) in an utterance may not represent the utterance-level semantics compared to other generic phrases. Other methods for span extraction include syntactic chunking, but mostly require supervised data [137] and heuristics (such as considering "noun phrases" or "verb phrases"), and thus are not flexible and robust compared to our method.

Finally, target spans can be found in syntactic structures which can be potentially induced from supervised parsers or unsupervised grammar induction [138, 139, 140, 141, 142]. [143] probe LMs and observe that recursively splitting sentences into binary trees in a top-down approach can correlate to constituency parsing. However, unlike the task of predicting relationship between words in a sentence where phrases at each level of a hierarchical structure are valid, detecting clear boundaries is critical to span extraction but challenging with various phrase lengths. Even though more flexible compared to semantic parsers that are limited by pre-defined roles, there is no straightforward way to apply these methods to span extraction.

4.4 Methodology

Our proposed method for slot schema induction consists of a fully unsupervised span extraction stage followed by coarse-to-fine clustering.

4.4.1 Overview

Given user utterances from raw conversations, our goal is to induce the schema of slot types $\mathcal{S}$ and their corresponding slot values. The span extraction stage extracts spans (e.g., "with wifi") from an utterance $\mathbf{x}$. The candidate spans from all user utterances are then clustered into a set of groups $\mathcal{S}$ where each group s_i corresponds to a slot type such as "internet" with values "with wifi", "no wifi", and "doesn't matter". The induced slot schema can be later used for downstream applications such as dialog state tracking and response generation.

4.4.2 Candidate span extraction

Challenges Since it is unclear what spans are meaningful phrases representative of task-specific slots, candidate span extraction presents two challenges. Firstly, with either supervised or unsupervised predicted structures, there is no protocol on what constituent and from what level we should extract the spans from without relying on dataset-specific heuristics, especially as structured representations are often compositional [144]. The second challenge is that span extraction methods should be flexible and robust to unseen tasks and domains. To tackle these problems, we leverage pre-trained LMs and propose a novel bottom-up attention-based span extraction method regularized by unsupervised PCFG for better structure representation. Because our method does not need any supervised data, the second problem can be effectively addressed by in-domain self-training. The full algorithm is outlined in Algorithm 1.

Algorithm 1: Span Extraction

Require: $\mathbf{x} = x_1, x_2, \ldots, x_n$: a user utterance $\mathbf{x}$

$\mathbf{t} \leftarrow PCFG(\mathbf{x})$ {A Chomsky normal form (binary) tree structure from self-supervised PCFG}
$\mathbf{a} \leftarrow LM(\mathbf{x})$ {Attention distribution from a LM}
$\mathbf{d} \leftarrow [f(a_i, a_{i+1})$ for $i = 1, 2, \ldots, n-1]$ {Distance between consecutive tokens using a distance function f}
$\tau \leftarrow$ median($\mathbf{d}$)
sort $\mathbf{d}$ in increasing order, d_i still represents $f(a_i, a_{i+1})$
for all d_i in $\mathbf{d}$ **do**
    if $d_i < \tau$ and using PCFG **then**
        if $node_i$ and $node_{i+1}$ are siblings in PCFG **then**
            $node_{i+1} \leftarrow \{node_i, node_{i+1}\}$ {merge nodes, assign new parents}
        end if
    else if $d_i < \tau$ **then**
        $w_{i+1} \leftarrow \{w_i, w_{i+1}\}$ {merge two tokens}
    end if
end for

Bottom-up attention-based extraction with LMs and PCFG regularization Recent studies reveal that attention distributions in pre-trained LMs can indicate syntactic relationships among tokens [145]. Therefore, we hypothesize that similar attention

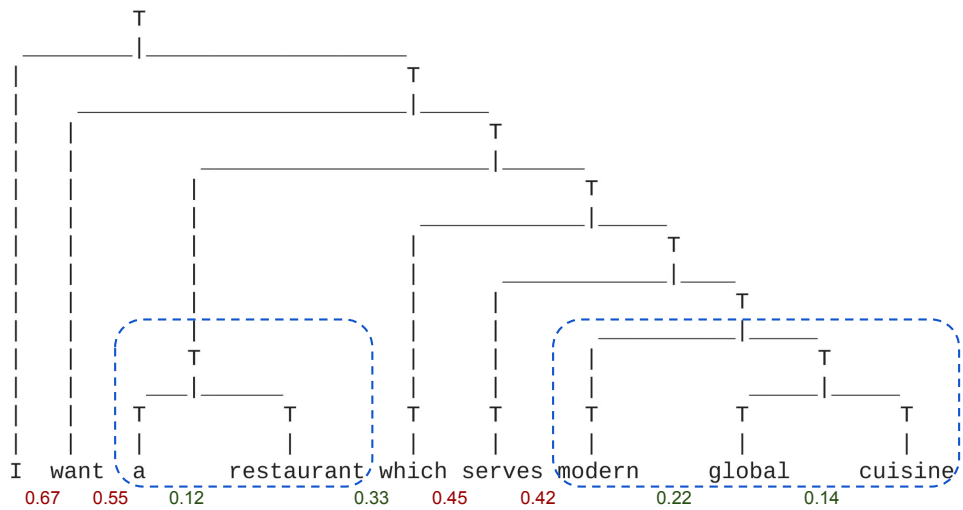

Figure 4.2. Illustration of span extraction where LM-derived distance function (distances between tokens are shown below the text) is constrained by a structure predicted by PCFG (tree structure shown in the figure). Numbers in red are above the median threshold (0.375) while numbers in green are below, indicating that the tokens share similar semantics and are from the same span. We can then extract candidate phrases "a restaurant" and "modern global cuisine", together with unigrams "I", "want", "which", and "serves".

distributions indicate tokens to form a meaningful phrase. We define the distance between attention distributions as a symmetric Jensen-Shannon divergence [145], and iteratively merge tokens whose distance is smaller than a threshold[2] in a bottom-up fashion. We start from the smallest distance to the largest, where the merged tokens are considered as a new token in the next iteration but the distribution distance with adjacent tokens remains the same. Fig. 4.2 illustrates the distances between tokens from a pre-trained LM for an example sentence where adjacent tokens such as "global" and "cuisine" are merged but not "serves" and "modern". This new decoding method enables us to effectively group tokens into phrases with precise boundaries.

Although LMs can be used to induce grammar, their training objectives are not optimized for sentence structure prediction, hence falling behind unsupervised PCFG [143] on syntactic modeling. Utilizing attention distribution from LM representations to extract spans can thus be fuzzy and noisy. We therefore employ unsupervised PCFG proposed by [146] as a mechanism to regularize our bottom-up span extraction. Instead of relying

[2]We use the median of all pairwise distances in an utterance in the experiments. We also compared other thresholds such as mean but did not observe significant difference.

solely on attention distribution, we in addition require two tokens to share the the same parent in the predicted PCFG tree structure before merging. This extra requirement reduces the noise from the distribution divergence in a sub-optimal structure representation. An example illustrating the necessity of span constraint is given in Fig. 4.2. Even though the distance between "restaurant" and "which" (0.33) is small, we disregard this span since they do not belong to the same parent in the PCFG structure. After merging two tokens, we assign the grandparent of the two tokens as the new parent, and continue the iteration until all distances are examined.

Self-supervised in-domain training Our attention-based approach enables us to extract phrases beyond certain n-grams, or certain types of phrases in a specific hierarchical layer. More importantly, it is appealing to adapt to new domains, where a LM can be further trained to encode structure representations without any annotated data and to group tokens into candidate phrases based on the training corpus. To encourage efficient span extraction above token-level representation, we further pre-train a SpanBERT model [147] by predicting masked spans together with a span boundary objective (denoted as TOD-Span) on TOD data [148]. This process can be thought of as incorporating corpus statistics such as phrase frequency into the model implicitly [149].

The unsupervised PCFG is trained to maximize the marginal likelihood of in-domain utterances with the inside-outside algorithm on the same TOD dataset. Similar to self-supervised LMs, this process is flexible and robust against domain mismatch, a common problem with supervised parsers [150]. At inference time, the trained model predicts a Chomsky normal form from Viterbi decoding [151].

4.4.3 Clustering candidate spans

Challenges After extracting candidate spans as potential slot values, we apply contextualized clustering on them to form latent concepts each slot value belongs to. We face two major challenges. Firstly, for any clustering method, hyperparameters such as the number of clusters are critical to the clustering quality, while they are not known for a new domain. Secondly, because of the trivial differences in slot types (for example, a location can be a "train departure place", or a "taxi arrival place"), clustering requires considering

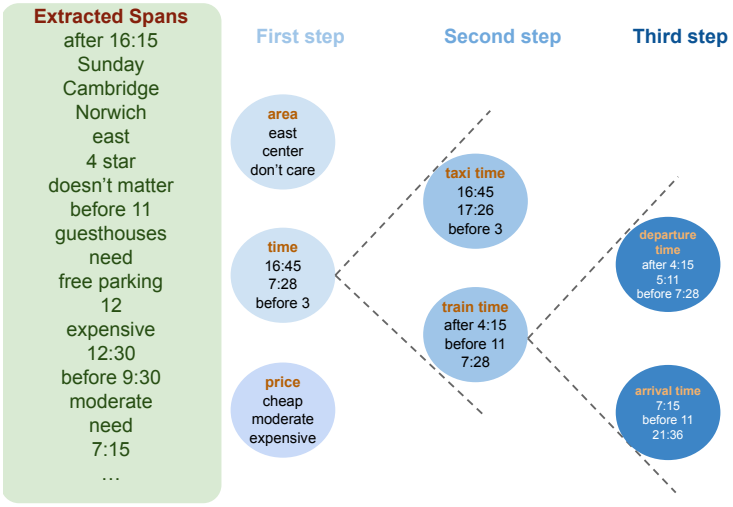

Figure 4.3. Multi-step clustering procedure. Each coarse cluster is further refined by next-step clustering. The first step uses contextualized span representations to capture salient groups (such as a cluster about time), and the second step uses the utterance-level representations of each span to capture domain and intent information. The third step utilizes span-level representation for fine-grained slot types.

different dimensions of semantics and pragmatics. To address these problems, we propose an auto-tuned, coarse-to-fine multi-step clustering method.

Auto-tuning hyperparameters To avoid hyperparameter tuning, we utilize density-based HDBSCAN [152]. Compared to other clustering methods such as K-Means, HDBSCAN is mainly parametrized by the minimum number of samples per cluster, and resulting clusters are known to be less sensitive to this parameter. We set this number automatically by maximizing the averaged Silhouette coefficient [153]

$$s = \frac{b - a}{max(a, b)}$$

across all clusters where a represents the distance between samples in a cluster, and b measures the distance between samples across clusters.

Multi-step clustering The input to our first-step clustering is the contextualized span-level representation from the extracted spans. To prevent the surface-level token embeddings from playing a dominant role, we replace candidate spans with masked tokens and use the contextual representation of the masked spans [154].

[155] suggest that we may only identify salient clusters (e.g., cardinal numbers), but cannot separate for example, different types of cardinals (e.g., number of people or number

of stays). Thus, in the second step, we cluster examples within each cluster from the first step leveraging utterance level representation of spans (i.e. the [CLS] token of the utterance where the span is from). This enables us to distinguish between domains and intents as they reflect utterance-level semantics. For example, we may find a cluster of time information (e.g., "11 AM") in the first step, and the second step clustering is to differentiate between train and taxi booking time. Lastly, we cluster groups developed from the second step into more fine-grained types using span-level representations similar to the first step. After this multi-step clustering, we can potentially separate for instance, departure time and arrival time in train booking. This process is illustrated in Fig. 4.3. Each cluster represents a slot type, with slot values shown as data points. This multi-step clustering brings an additional benefit of inducing the slot schema with hierarchy, where sub-groups in further steps belong to the same parent group.

4.5 Experiments

To examine the quality of our induced schema, we perform *intrinsic* and *extrinsic* evaluations. Our intrinsic evaluation compares the predicted schema with the ground truth schema by measuring their overlap in slot types and slot values. This indicates how well our induced schema aligns with the expert annotation. The extrinsic evaluation estimates the usefulness of the induced schema for downstream tasks, for which we consider dialog state tracking and response generation tasks. Experiments are conducted on MultiWOZ [115] and SGD [122] datasets following previous research. We also apply and evaluate our method for both intent and slot schema induction on realistic scenarios (See Section 4.6).

Baselines We compare our proposed approach with different setups against DSI [117], which uses supervised tools and heuristics. We evaluate different span extraction methods including using parsers only, leveraging distance functions from LMs, and combining LMs with unsupervised PCFG. Specifically, NP extracts all noun phrases[3], DSI cand. uses the same candidates phrases as DSI, and PCFG and CoreNLP [156] extract phrases from an unsupervised and supervised structure respectively by taking the smallest constituents

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

above the leaf level. These baselines solely rely on parsers. For our bottom-up attention-based LM methods (Section 4.4.2), we compare spans extracted using representations from BERT [7], SpanBERT [147], TOD-BERT [148], and our span-based TOD pre-training from masking random spans (TOD-Span). Lastly, we combine the LMs with unsupervised PCFG structures.

4.5.1 Slot schema induction

To evaluate the induced schema against ground truth, we need to match clusters to ground truth labels[4]. Previous work on dialog schema induction either requires manual mapping from a cluster to the ground truth [120] or compares predicted slot values to its state annotation at each turn [117]. These can create noises and biases, hence not practical when no annotation is available. Particularly, [117] compare candidate spans to corresponding reference slot types at each turn, which is a small subset of the ground-truth ontology. This would overestimate the performance of schema induction since the matching is more evident and is different from defining schemas in realistic settings. Instead, we simulate the process of an expert annotator mapping clusters to slot names by considering the general contextual semantics of spans in a cluster.

Setup We consider semantic representations of ground truth clusters as labels. Specifically, we calculate the contextual representation of spans averaged across all spans in an induced cluster as cluster representations, and compare that with ground truth slot type representations computed in the same way. For fair comparison among different methods, we use BERT to obtain span representations. We assign the name of the most similar slot type representation to a predicted cluster measured by cosine similarity. If the score is lower than 0.8 [117], the generated cluster is considered as noise without mapping, which simulates when a human cannot label the cluster. We report precision, recall, and F1 on the induced slot types. When the number of clusters is larger than the ground truth, multiple predicted clusters can be mapped to one slot type. This evaluation process is identical to human annotation, but may be biased towards more clusters. Thus we report

[4]Predicting labels for each cluster is out of the scope of this paper. Since there are many ways to assign labels with equal semantics to a cluster (e.g., "food" vs. "restaurant type")

method	# clusters	slot type	slot value
Baseline			
DSI	522	87.72	37.18
Parser only			
NP	88	69.39	47.46
DSI cand.	113	85.19	49.71
PCFG	339	91.53	53.62
CoreNLP	292	87.72	54.43
Language model only			
BERT	340	85.71	55.80
SpanBERT	343	89.66	45.21
TOD-BERT	219	89.66	50.89
TOD-Span	374	85.71	55.29
Language model contrained on unsupervised PCFG			
BERT	350	87.72	52.32
SpanBERT	203	89.66	44.51
TOD-BERT	245	91.53	48.13
TOD-Span	290	**96.67**	**58.71**

Table 4.1. Schema induction results on MultiWOZ. TOD-Span (span-based LM further pre-trained on in-domain data) regulated by PCFG achieves the best performance on slot type induction and slot value induction evaluated by F1 scores. All methods (except DSI) differ only by span extraction (i.e., same clustering).

the number of induced clusters for reference. Similarly, within each slot type, we compute the overlapping of cluster values to all ground truth slot values and report precision, recall, and F1 by fuzzy-matching scores [117], averaged across all types.

Results Table 4.1 shows the results of schema induction on slot types and slot values. All methods lead to a number of clusters within a similar range (except the slightly larger 522 clusters for DSI), indicating that the results are not biased and are comparable. When the candidate span input to our proposed multi-step clustering is the same as the baseline DSI using POS tagging and coreference (DSI cand.), we achieve similar performance on

method	turn level	joint level
Baseline		
DSI	18.29	25.22
Parser only		
PCFG	25.43	32.39
Language model only		
BERT	24.35	30.18
SpanBERT	20.24	26.07
TOD-BERT	25.05	34.94
TOD-Span	29.72	38.89
Language model contrained on unsupervised PCFG		
BERT	23.27	30.09
SpanBERT	20.96	27.25
TOD-BERT	27.11	31.92
TOD-Span	**39.59**	**46.69**

Table 4.2. DST results on MultiWOZ. We show F1 scores of turn and joint level. TOD-Span regularized by PCFG achieves the best performance.

slot type induction (85.19) and better results on slot values (49.71). This illustrates the effectiveness of our proposed clustering method since the only difference from the DSI baseline is clustering. Compared to methods leveraging noun phrases (NP), or supervised parsers (CoreNLP), using an unsupervised PCFG trained on in-domain TOD data can achieve comparable or superior results.

If we extract spans using LMs only, different models perform similarly on both slot types and slot values. However, when regularized by an unsupervised PCFG structure, we observe a large performance boost especially with TOD-Span. This indicates that the unsupervised PCFG can provide complementary information to LMs. In addition, results show that further pre-training a LM at span level is more efficient.

belief state	BLEU
None	15.6
DSI	13.9
TOD-Span + PCFG	16.4
Ground truth	17.9

Table 4.3. Response generation results on MultiWOZ. Our method introduces positive inductive bias.

4.5.2 Application in DST

Now that we have mapped induced clusters to ground truth names, we can immediately evaluate DST performance by identifying slot values and types as described above. This can be considered as a zero-shot setting.

Setup Following [117], we calculate the overlapping of the predicted slots and values with their corresponding ground truth at both the turn level and the joint level. At each turn, a fuzzy matching score is applied on predicted values [122] whose corresponding slot types are in the ground truth. On the other hand, even if a slot value is predicted correctly but its slot type does not match the ground truth, no reward is accredited. On the joint level, we calculate the score for accumulative predictions up to the current turn.

Results Table 4.2 summarizes the results for DST. Similar to the trend in schema induction, constraining an in-domain fine-tuned LM (TOD-Span) on an unsupervised structure representation (PCFG) achieves the best performance (39.95 on turn level), significantly outperforming a strong baseline DSI (18.29)[5]. We also note that because all accumulated predictions are evaluated for partial rewards instead of exact matching on all slot types in standard DST evaluation, the joint level scores are higher than the turn level from accumulative scores.

4.5.3 Application in response generation

The above settings map latent slot clusters to ground truth analogous to expert designs so that we can evaluate the alignment with human annotations. This experiment investigates whether the induced latent schema is still useful before mapping.

[5]We use their provided data and model to run the DSI baseline. The reason the score is lower here than their report is due to slot type matching (Section 4.5.1).

Setup We modify the model of [157, 158] by appending the predicted labels (i.e., cluster index such as "10-24" indicating a specific slot type) and values to the context (e.g., "I need a train at 7:45. [10-24] 7:45" as input). The added belief state can be considered as a prior to generate responses similar to [159]. Since we do not have the mapped names of the slots, we only report the BLEU score rather than other metrics used in response generation that require entity-level matching (e.g., inform rate). This is a more practical setting directly evaluating on the induced schema compared to previous work [117], where dialog act is modeled with delexicalized input utterances ([160], not feasible because ontology is required from a pre-defined schema for delexicalization).

Results Table 4.3 compares the performance of using no belief state (None), belief state induced by DSI, our introduced method (TOD-Span + PCFG), and ground truth. Results show that our induced schema introduces a positive inductive bias (16.4) compared to the baseline (15.6) and is close to the ground truth schema with actual slot type names. We conjecture that the lower performance of DSI is due to the larger number of latent types (522) which creates noises in model training. Thus, our induced slot schema is useful for downstream applications.

4.6 Analysis

Comparison among different methods Our results show that in general, span-based pre-training methods outperform token-based, and continued pre-training on in-domain data is important. When regularized by PCFG structures, we observe a large performance boost on TOD-BERT and TOD-Span, however the PCFG structure does not help BERT and SpanBERT when the LM is trained on general domain data only. We speculate that the LM representation trained on generic text is not compatible with the predicted structure induced via in-domain self-supervision. In addition, we believe that the performance gap between our proposed method and previous research using rules from supervised parsers (such as NPs and coreference) is larger when the data is less biased (for example, if NP is not dominant as slot values, [161]). Moreover, our proposed method is data-driven, indicating that the slots are determined by the dialog corpus. If there are specific

annotation requirements, we can inject inductive bias to the LM to change distribution distances [162] or add rules to incorporate such conditions.

Comparison among different datasets On MultiWOZ, our method induces 30 out of 31 slot types in the ontology except "hospital-department", which only appears once in the dialog corpus. For slot values, errors are mostly from low precision due to loose boundaries and semantic matching (e.g., predicting "free wifi", and "include free wifi", where the target value is "yes"). In comparison, DSI induces 26 slot types, with similar slots mixed (such as mapping "taxi-arriveby" to "taxi-leaveat"). It receives a relatively low slot value score since spans extracted using rules are not robust and compatible. On SGD where 82 slot types are defined in the ontology, our method induces 50 and DSI induces 72. The main reason for this low recall is similar slot types with overlapping values (such as "media-genre" and "movies-genre"), and single-value slots (such as "has-wifi" with the value "True"). More importantly, SGD has a smaller utterance length, making it more difficult to map to the correct slot type without considering more context. With a magnitude more number of clusters, DSI (11992 clusters) has a higher chance to map predicted slots to target slot types which explains better performance than ours on schema induction. However, this large number of clusters make it infeasible for humans to use, and our induced schema is comparable in downstream tasks such as DST.

We also apply our method on internal customer data for both intent (by applying multi-step clustering directly on utterances) and slot schema induction. Compared to MultiWOZ and SGD, schema in more realistic scenarios is more complicated and the slot boundaries are less clear. Nevertheless, our method is still effective in inducing the majority of the schema to find intents such as "change password" and slot types such as "devices". We observe similar findings on the Ubuntu dialog corpus [163].

Ablation studies Table 4.4 illustrates the performance comparisons with different numbers of clustering steps, as well as input representations. Results demonstrate that compared to one-step (using masked span representation) and two-step (adding utterance representation), our three-step clustering method induces a more fine-grained schema, which is more effective for downstream tasks. The number of steps can be customized to

		schema		DST	
method	# clusters	type	value	turn	joint
Different number of clustering steps					
one-step	31	60.87	39.74	23.58	30.68
two-step	99	83.64	46.66	35.21	41.94
Original representation instead of masked					
unmasked rep.	284	85.71	53.30	27.93	36.40
Three-step masked clustering					
Three-step masked	290	96.67	58.71	39.59	46.69

Table 4.4. Ablation results on MultiWOZ with TOD-Span constrained on PCFG. Using masked presentation for multi-step clustering improves the performance on schema induction and DST by a large margin.

real use cases depending on target granularity[6]. In addition, if we use the original input rather than the masked phrase representation, the performance drops by a large margin (85.71 on slot type). This suggests that the surrounding context is more critical than the surface embeddings for schema induction, especially when the same phrase can serve different functions even in the same domain (such as locations).

DST Error analysis Suggested by the relatively high span extraction accuracy (68.13 F1 score), we find that the majority of the problems in DST come from cluster mapping. This is caused by either excessive surrounding information or by the lack of context from previous turns. For instance, in the utterance "Can I book it for 3 people", the "3 people" can be mapped to either "restaurant-book people" or "hotel-book people", since we extract the contextual information from the current turn only. If more context is considered, the mapping performance including results on downstream tasks is expected to improve. Another issue is with span boundary. Even though we apply fuzzy matching, the evaluation still penalizes correct predictions (such as "indian food") from its ground truth ("indian"), since we do not have training signals to identify the target boundaries.

[6]More steps were also conducted but we observed lower Silhouette coefficient and lower quality in preliminary studies.

4.7 Summary

In this chapter, we introduce methods to represent more fine-grained low-level attributes such as ontology. Illustrated with an example in the task-oriented dialog domain, we propose a fully unsupervised method for slot schema induction where we propose to learn attributes such as phrases that are critical to the target task. Compared to previous research, our method can be easily adapted to unseen domains and tasks to extract target phrases before clustering into fine-grained groups without domain constraints. We conduct extensive experiments and show that our proposed approach is flexible and effective in generating accurate and useful schemas without task-specific rules in both academic and realistic datasets. We believe that our method could also be applied to other languages (since no supervised parser is required) and tasks where the target is not explicitly annotated [164]. One immediate next step would be to extend our method to represent more complex structures.

Chapter 5
Task-specific Attribute Representation

5.1 Overview

Different from high-level and low-level attributes, there are cases where it might be easier to consider the task itself as an attribute, instead of implicitly learning some intermediate representation to connect some trained parameters to the output space. Different from previous chapters, methodologies to represent task-specific attributes are not universal, as each task may require different optimal modeling methods. In this chapter, we choose two distant task-specific attributes for illustration.

We first introduce example representation, where examples are considered as attributes of spans or sentences. Readers may draw some connection to low-level attributes where we represent spans and ontolgoies in Chapter 4. The main difference is that instead of learning and extracting the attribute representation implicitly, we can view example representation as the output space directly by disentangling from the target labels. Then we introduce problem representation, where the we consider the mostly critical problems in neural language models including toxicity and inconsistency that are inherently learned from model pre-training. Although problems can be viewed as high-level attributes as examined in Chapter 3, we study representing problems as attributes directly, instead of learning hidden representations towards controlling neural models.

This chapter is based on our works [132, 165] that were published at the NAACL 2021

and EMNLP 2021 conference, which I lead as the main author. Luheng He, Yuan Zhang, Xinya Du, Panupong Pasupat, and Qi Li (from Google), and Kenji Sagae serve in an advisory capacity.

5.2 Example Representation for Retrieval-based Language Understanding

5.2.1 Introduction

Few-shot learning is challenging due to the imbalance in the amount of data between the source and target domains. Traditional classification methods, even with the recent advancement of pre-trained language models [6, 7], could suffer from over-fitting [166, 167] or catastrophic forgetting [168] when incorporating the data-scarce target domain. On the other hand, *metric learning methods* [169, 170, 166] have been shown to work well in few-shot scenarios. These methods are based on modeling similarity between inputs, effectively allowing the model to be decoupled from the semantics of the output space. For example, a model would learn that the utterance *"I'd like to book a table at black horse tavern at 7 pm"* (from Figure 5.1) is similar to *"make me a reservation at 8"* and thus are likely to have similar semantic representations, even without knowing the semantic schema in use. Unlike learning output labels, which is difficult when examples are scarce, learning a similarity model can be done on the abundant source domain data, making such models data-efficient even in few-shot settings.

We focus on *retrieval-based* methods among other metric learning methods. The most basic setting of retrieval-based model for few-shot learning is: after training a similarity model and encoding target domain data into the index, we can retrieve examples most similar to the given input, and then make a prediction based on their labels. Compared to methods that do not maintain an index, such as Prototypical Networks [166], retrieval-based methods are less sensitive to outliers with few data points, and are powerful when we have abundant data in the source domain [167]. However, applying retrieval-based models on tasks with a structured output space is non-trivial. For example, even if we know that the utterance in Figure 5.1 is similar to *"make me a reservation at 8"*, we cannot

directly use its slot values (e.g., the `time` slot has value *"8"* which is not in the input), and not all slots in the input (e.g., *"black horse tavern"*) have counterparts in the retrieved utterance. While previous works have exploited token-level similarity methods in a BIO-tagging framework, they had to separately simulate the label transition probabilities, which might still suffer from domain shift in few-shot settings [171, 172].

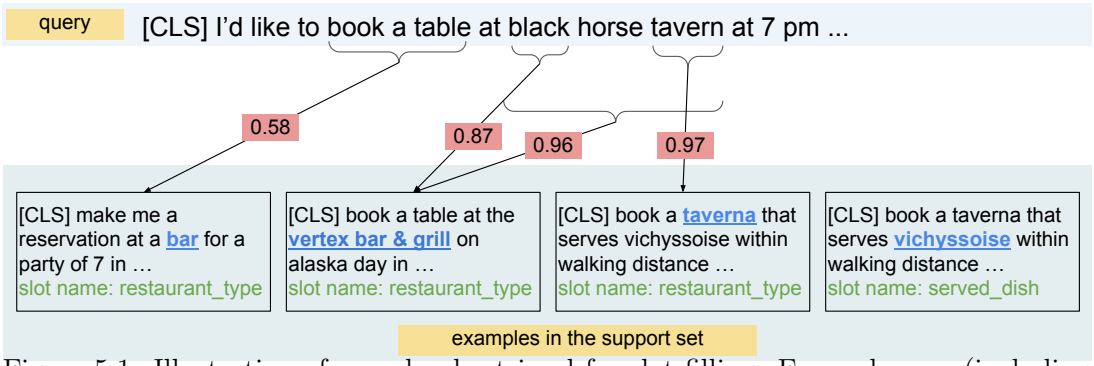

Figure 5.1. Illustration of span-level retrieval for slot filling. For each span (including spans that are not valid slots such as *"book a table"*) in the input utterance, we retrieve its most similar span from the retrieval index, and then assign the slot name as the prediction with a similarity score. We use modified beam search to decode an output that maximizes the average similarity score. The gold slots are *"black horse tavern"* and *"7 pm"* in this example.

5.2.2 Setup

We consider two tasks where the input is an utterance $\mathbf{x}$ with tokens $x_1, \ldots, x_n$ and the output is some structure $\mathbf{y}$. For the *slot filling* task, the output $\mathbf{y}$ is a set of non-overlapping labeled spans $\{(r_i, \ell_i)\}_{i=1}^m$ where r_i is a span of $\mathbf{x}$ (e.g., *"7 pm"*) and ℓ_i is a slot name (e.g., `time`). For the *intent classification* task, the output $\mathbf{y}$ is simply an intent label ℓ for the whole utterance $\mathbf{x}$. For notational consistency, we view intent classification as predicting a labeled span (r, ℓ) where $r = \mathbf{x}_{1:n}$.

In the few-shot setup, examples $(\mathbf{x}, \mathbf{y})$ are divided into *source* and *target* domains. Examples in the target domain may contain some labels ℓ that are unseen in the source domain. The model will be given ample training data from the source domain, but only a few training examples from the target domain. For instance, the model receives only $K = 5$ examples for each unseen label. The model can be evaluated on test data from both domains.

5.2.3 Model

We propose a retrieval-based model, `Retriever`, for intent classification and slot filling in the few-shot setting. Figure 5.1 illustrates our approach. At a high level, from examples $(\mathbf{x}, \mathbf{y})$ in the target training data (and optionally the source training data), we construct a retrieval index consisting of labeled spans (r, ℓ) from $\mathbf{y}$. Given a test utterance $\mathbf{x}$, for each span of interest in $\mathbf{x}$ (all spans $\mathbf{x}_{i:j}$ for slot filling; only $\mathbf{x}_{1:n}$ for intent classification), we retrieve the most similar labeled spans (r, ℓ) from the index, and then use them to decode an output $\mathbf{y}$ that maximizes the average span similarity score.

The use of retrieval provides several benefits. For instance, we empirically show in Section 5.2.5.1 that the model does not suffer from catastrophic forgetting because both source and target data are present in the retrieval index. Class imbalance can also be directly mitigated in the retrieval index. Additionally, since the trained model is non-parametric, we could replace the retrieval index to handle different target domains without having to retrain the model. This also means that the model does not need access to target data during training, unlike traditional classification methods.

5.2.3.1 Retriever

The retriever is the only trainable component in our model. Given a query span $r' = \mathbf{x}_{i:j}$ from the input $\mathbf{x}$, the retriever returns a set of labeled spans (r, ℓ) with the highest similarity scores $s(\mathbf{z}, \mathbf{z}')$, where $\mathbf{z} = E(r)$ and $\mathbf{z}' = E(r')$ are the contextualized embedding vectors of r and r', respectively.

Similarity score To compute the contextualized embeddings $\mathbf{z}$ and $\mathbf{z}'$ of spans r and r', we first apply a Transformer model initialized with pre-trained BERT on the utterances where r and r' come from. For slot filling, we follow [173] and define the span embedding as the concatenated embeddings of the its first and last wordpieces. For intent classification, we use the embedding of the [CLS] token. We then define $s(\mathbf{z}, \mathbf{z}')$ as the dot product between $\mathbf{z}$ and $\mathbf{z}'$.

Training with batch softmax We use examples from the source domain to train `Retriever`. Let $\ell_1, \ldots, \ell_N$ be the N class labels (slot or intent labels) in the source domain. To construct a training batch, for each class label ℓ_i, we sample B spans $r_i^1, \ldots, r_i^B$ from

the training data with that label, and compute their embeddings $\mathbf{z}_i^1, \ldots, \mathbf{z}_i^B$. Then, for each query span r_i^j, we compute similarity scores against all other spans in the batch to form a $B \times N$ similarity matrix:

$$S_i^j = \begin{bmatrix} s(\mathbf{z}_i^j, \mathbf{z}_1^1) & s(\mathbf{z}_i^j, \mathbf{z}_2^1) & \ldots & s(\mathbf{z}_i^j, \mathbf{z}_N^1) \\ s(\mathbf{z}_i^j, \mathbf{z}_1^2) & s(\mathbf{z}_i^j, \mathbf{z}_2^2) & \ldots & s(\mathbf{z}_i^j, \mathbf{z}_N^2) \\ \vdots & \vdots & \ddots & \vdots \\ s(\mathbf{z}_i^j, \mathbf{z}_1^B) & s(\mathbf{z}_i^j, \mathbf{z}_2^B) & \ldots & s(\mathbf{z}_i^j, \mathbf{z}_N^B) \end{bmatrix}. \tag{5.1}$$

We now summarize the score between r_i^j and each label $\ell_{i'}$ by applying a reduction function (defined shortly) along each column to get a $1 \times N$ vector:

$$\hat{S}_i^j = \begin{bmatrix} s(\mathbf{z}_i^j, \mathbf{z}_1^*) & s(\mathbf{z}_i^j, \mathbf{z}_2^*) & \ldots & s(\mathbf{z}_i^j, \mathbf{z}_N^*) \end{bmatrix} \tag{5.2}$$

We use the softmax of $\hat{S}_i^j$ as the model's probability distribution on the label of r_i^j. The model is then trained to optimize the cross-entropy loss on this distribution against the gold label ℓ_i.

We experiment with three reduction functions, *mean* (Eq. 5.3), *max* (Eq. 5.4), and *min-max*:

$$s(\mathbf{z}_i^j, \mathbf{z}_{i'}^*) = \frac{1}{B} \sum_{j'=1}^{B} s(\mathbf{z}_i^j, \mathbf{z}_{i'}^{j'}) = s\left(\mathbf{z}_i^j, \frac{1}{B} \sum_{j'=1}^{B} \mathbf{z}_{i'}^{j'}\right) \tag{5.3}$$

$$s(\mathbf{z}_i^j, \mathbf{z}_{i'}^*) = \max_{\substack{1 \le j' \le B; \\ j' \ne j \text{ if } i=i'}} s(\mathbf{z}_i^j, \mathbf{z}_{i'}^{j'}) \tag{5.4}$$

The *mean* reduction averages embeddings of the spans with the same label and is equivalent to Prototypical Networks. Similar to hard negative sampling to increase margins among classes [174, 175, 176], *max* takes the most similar span to the query (excluding the query itself) as the label representation, while *min-max* takes the least similar span when considering spans with the same label as the query.

5.2.3.2 Inference

After training, we build a dense retrieval index where each entry (r, ℓ) is indexed by $\mathbf{z} = E(r)$. The entries (r, ℓ) come from examples $(\mathbf{x}, \mathbf{y})$ in the *support set* which, depending

on the setting, could be just the target training data or a mixture of source and target data. For each query span r' of the input utterance $\mathbf{x}$, we embed the span and compute the similarity scores against all index entries.

Intent classification For intent classification, both index entries and query spans are restricted to the whole utterances. The entire process thus boils down to retrieving the most similar utterance based on the [CLS] token embedding. We simply output the intent label of the retrieved utterance.

Slot filling In contrast to BIO decoding for token-level similarity models [172], decoding with span retrieval results poses unique challenges as gold span boundaries are not known a priori. Hence, we use a modified beam search procedure with simple heuristics to compose the spans. Specifically, for each of the $n \times m$ spans in an utterance of length n, we retrieve the most similar span from the retrieval index. Then we normalize the similarity scores by L2-norm so that they are within the range $[0, 1]$. Since we do not explicitly predict span boundaries, all $n \times m$ spans, including non-meaningful ones (e.g., *"book a"*), will have a retrieved span. Such non-meaningful spans should be dissimilar to any labeled span in the retrieval index. We thus choose to filter the spans with a score threshold to get a smaller set of candidate spans, and use beam search to decode a set of spans with maximum average scores.[1] We go through the list of candidate spans in the descending order of their similarity scores. For each candidate span, we expand beam states if the span does not overlap with the existing spans in the beam. The search beams are pruned based on the average similarity score of the spans included so far.

5.2.4 Experiments and Results

We evaluate our proposed approach on two datasets: CLINC [177] for intent classification and SNIPS [178] for slot filling. Note that we use *max* (Eq. 5.4) as the reduction function for both tasks since it empirically yields the best results. The effect of reduction functions will be analyzed later in Section 5.2.5.1.

[1] We use beam search for simplicity. Other search methods such as Viterbi algorithm [151] can also be used.

	support_set=all			support_set=balance			support_set=tgt
	tgt	src	avg	tgt	src	avg	tgt
Initial BERT							
Proto^{frz}	14.07	25.02	21.37	-	-	-	-
Retriever^{frz}	8.24	54.76	39.25	22.09	25.29	24.22	37.93
Pre-train on src domain							
BERT fine-tune	-	96.51	-	-	-	-	-
Proto	75.02	95.73	88.83	-	-	-	-
Retriever	62.69	**97.08**	85.62	75.93	95.44	88.94	88.53
Retriever min-max	66.00	96.64	86.43	71.82	95.14	87.37	86.38
Fine-tune on tgt domain							
BERT fine-tune	78.89	43.91	55.57	-	-	-	-
Proto	80.44	95.57	90.53	-	-	-	90.35
Retriever	66.76	96.95	86.89	79.20	95.50	90.07	**91.16**
Retriever min-max	67.64	96.84	87.11	77.60	95.35	89.43	89.56
Fine-tune on tgt domain with src data							
BERT fine-tune	72.00	95.18	87.45	-	-	-	-
Proto	83.33	94.82	90.99	-	-	-	90.22
Retriever	69.51	97.04	87.86	**84.95**	95.41	**91.92**	90.78
Retriever min-max	71.35	96.96	88.42	81.00	94.55	90.03	89.82

Table 5.1. Intent accuracy on CLINC for $n_c = 10, n_i = 10$ with 5-shots. Our retrieval-based method outperform BERT fine-tune and Prototypical Networks in both target and source domains. We report results for our method when the support set consists of all examples in the source and target domains (all), when the support set consists of balanced few-shot number of examples for intents in both source and target domains (balance), and when the support set consists of examples of the target domain only (tgt) which serves as an upper-bound.

5.2.4.1 Intent Classification

The CLINC intent classification dataset [177] contains utterances from 10 intent categories (e.g., "travel"), each containing 15 intents (e.g., "flight_status", "book_flight"). To simulate the few-shot scenario where new domains and intents are introduced, we desig-

nate n_c categories and n_i intents per category as the source domain (with all 100 training examples per intent), and use the remaining $150 - n_c \times n_i$ intents as the target domain. We experiment with $(n_c, n_i) = (10, 10)$, $(8, 10)$, and $(5, 15)$. The target training data contains either 2 or 5 examples per target intent.

We compare our proposed method `Retriever` with a classification model `BERT fine-tune` and a Prototypical Network model `Proto`. The former learns a linear classifier on top of BERT embeddings [7], and the latter learns class representations based on Prototypical Networks.[2] We also show results with the initial BERT checkpoint without training ($\texttt{Proto}^{\texttt{frz}}$, $\texttt{Retriever}^{\texttt{frz}}$). We use the same batch size for all models, and tune other hyperparameters on the development set before testing.

Evaluation We sample domains and intents three times for each (n_c, n_i) setting, and report average prediction accuracy. We report accuracy on intents from the target domain (tgt), source domain (src), and the macro average across all intents (avg).

Moreover, we evaluate the models with the following support set variations: with target domain data and all data in the source domain (support_set=all), with equal number of examples (same as the few-shot number) per intent (support_set=balance), and with only examples from the target domain (support_set=tgt). The last one serves as an upper-bound for the target domain accuracy.

Results Table 5.1 shows the results for $(n_c, n_i) = (10, 10)$ and 5 examples per target intent; results on other settings exhibit the same patterns. We observe that `Retriever` performs the best on the source domain (97.08%) before fine-tuning. `Retriever` also achieves the highest accuracy on the target domain (84.95%) after fine-tuning, while maintaining competitive performance on the source domain (95.41%) among all the methods.

5.2.4.2 Slot Filling

SNIPS [178] is a slot filling dataset containing 39 slot names from 7 different domains: GetWeather (GW), PlayMusic (PM), AddToPlaylist (ATP), RateBook (RB), FindScreeningEvent (FSE), BookRestaurant (BR), and SearchCreativeWork (SCW). Following [172], we train

[2]Previous work show that Prototypical Networks outperforms other optimization-based and metric-learning models such as MAML in (intent) classification tasks [167, 179].

	GW	PM	ATP	RB	FSE	BR	SCW	Average F1
Classification-based								
BERT Tagging	59.41	42.00	46.07	20.74	28.20	67.75	58.61	46.11
Token-level								
SimilarToken^{frz}	53.46	54.13	42.81	75.54	57.10	55.30	32.38	52.96
MatchingToken	36.67	33.67	52.60	69.09	38.42	33.28	72.10	47.98
ProtoToken	67.82	55.99	46.02	72.17	**73.59**	60.18	**66.89**	63.24
L-TapNet+CDT+Proto	-	-	-	-	-	-	-	67.27
L-Proto+CDT^{pw}*	74.68	56.73	52.20	78.79	80.61	69.59	67.46	68.58
L-TapNet+CDT+Proto^{pw}*	71.64	67.16	75.88	84.38	82.58	70.05	73.41	75.01
Span-level (ours)								
Proto^{frz}	39.47	38.35	47.68	69.36	38.60	42.39	19.90	42.25
Proto	64.47	53.97	54.64	73.37	42.89	62.48	27.76	54.23
Retriever^{frz}	63.39	46.01	51.11	79.65	62.42	62.13	33.85	56.94
Retriever	**82.95**	**61.74**	**71.75**	**81.65**	73.10	**79.54**	51.35	**71.72**

Table 5.2. Results on SNIPS test data with 5-shot support sets. Our span-based retrieval model outperforms previous classification-based and token-level retrieval models even without label semantics. Classification-based and token-level results are reported in [172]. *Pair-wise embeddings (marked with ^{pw}) are expensive at inference time, so we do not compare our method with these directly.

models on five source domains, use a sixth one for development, and test on the remaining domain. We directly use the K-shot split provided by [172], where the support set consists of the minimum number of utterances such that at least K instances exist for each slot name. We also set $K = 5$ in our experiment.

We compare against two baselines and three models from the previous work. BERT Tagging is a BERT-based BIO tagging model [7] fine-tuned on the testing domain after training on the source domains, while SimilarToken^{frz} uses BERT embeddings to retrieve the most similar token based on cosine similarity without any training. MatchingToken and ProtoToken are two token-level methods that leveraged Matching Networks [170] and Prototypical Networks [166] respectively. L-TapNet+CDT+proto [172] is an adaptation

of TapNet [180] with label semantics, CDT transition probabilities, and Prototypical Networks.

We experiment with several variants of our proposed method. `Proto` trains Prototypical Networks to compute span class representations. `Retriever` retrieves the most similar slot example for each span. Both methods use the same decoding method. Similar to `SimilarToken`$^{\texttt{frz}}$, `Proto`$^{\texttt{frz}}$ and `Retriever`$^{\texttt{frz}}$ use the original BERT embeddings without any training. All models are trained on source domains and early stopped based on performance on the development domains.

Evaluation We report F1 scores for each testing domain in a cross-validation episodic fashion. Following [172], we evaluate each testing domain by sampling 100 different support sets and ten exclusive query utterances for each support set. We calculate F1 scores for each episode and report average F1 scores across 100 episodes.

Results Table 5.2 summarizes the experiment results on the SNIPS dataset. Our span-level method (`Retriever`) achieves higher averaged F1 than all five baselines, outperforming the strongest token-level method (`L-TapNet+CDT+proto`) by 4.45%. This shows that our model is effective at span-level predictions. More importantly, the better performance suggests that our span-level `Retriever` model is more efficient at capturing span structures compared to simulated dependencies as our method does not suffer from the potential discrepancy in the transition probabilities between the target and source domains.

Although [172] showed that adding pairwise embeddings with cross-attention yielded much better performance, this method is expensive both in memory and computation at inference time, especially when the support set is large [181]. For fair comparison, we do not directly compare with methods using pairwise embeddings (methods with $^{\texttt{pw}}$ in Table 5.2). Note that our method with pre-computed support example embeddings even outperforms `L-Proto+CDT`$^{\texttt{pw}}$ with less memory and computation cost.

5.2.5 Analysis

We conduct further discussion and ablation studies in this section to analyze the variations of different settings.

5.2.5.1 Intent Classification

Models without re-training The *pre-train on src domain* section in Table 5.1 shows the results of models that are only pre-trained on the source domains but **not** fine-tuned on the target domains. Classification models such as `BERT fine-tune` cannot make predictions on target domains in this setting. In contrast, even without seeing any target domain examples during training, retrieval-based models can still make predictions on new domains by simply including new examples in the support sets. With support_set=all, `Retriever` achieves 97.08% on the source domain while `Proto` performs worse than `BERT fine-tune`, consistent with previous findings [167]. `Retriever` achieves the best accuracy (75.93%) on target domains with a balanced support set on all intents (support_set=balance). More importantly, `Retriever` also achieves competitive accuracy on source domains (95.44%), demonstrating that our proposed model achieves the best of both worlds even without re-training on new domains.

Varying the support set at inference time The construction of the support set is critical to retrieval-based methods. In Table 5.1, we present the model performances under different support settings (all, balance, tgt). The support_set=tgt setting serves as an upper bound for the target domain accuracy for both `Retriever` and `Proto` methods. In general, `Retriever` achieve the best performance on the source domain intents when we use full support sets (support_set=all). In comparison, if we use a balanced support set (support_set=balance), we can achieve much higher accuracy on the target domain while having a slight degradation on the source domain intents prediction. This is because full support sets have more source domain examples to increase confusion over target domains.

Data for fine-tuning The *Fine-tune on tgt domain* section in Table 5.1 shows different model behaviors when fine-tuned on the target domain data directly. While `BERT fine-tune` achieves high accuracy (78.89%) on the target domain, it suffers from catastrophic forgetting on the source domain (43.91%). On the other hand, `Proto` and `Retriever` can get high accuracy on the target domain (80.44% and 79.20%) while maintaining high performance on the source domain.

When we combine data from the source domain, we observe performance gains in all

the models under the *Fine-tune on tgt domain with src data* section. Specifically, we add few-shot source domain examples as contrastive examples for the models to learn better utterance/class representations for `Retriever` and `Proto`. Results show that accuracy on the target domain increases by over 3% compared to only using target domain data. This suggests that unlike other retrieval-based methods such as kNN, `Retriever` does not require a large support set to guarantee prediction accuracy.

Impact of reduction functions We compare the reduction functions proposed in Section 5.2.3.1 and found that `max` performs the best. Since `mean` is equivalent to Prototypical Networks, we compared to `Proto` directly in the experiments. `min-max` is more intuitive in constrasting with least similar examples within the same class compared to `max`. However, its performance is worse than `max`. We speculate the reason to be that we retrieve the example with the maximum score at inference time so that the boundary margin may not be utilized.

5.2.5.2 Slot Filling

We note that `Retriever` outperforms strongest baselines but reaches a low score on the SCW domain. This may be due to the bigger difference between the test (SCW) and the development domain (GW) including the size of the support set and their respective slot names. For error analysis, we found that from all the correctly predicted slot spans, 96.73% predicted the correct slot names. This shows that the majority of the errors come from querying with invalid spans. We believe that span-based pre-training such as Span-BERT [147] could make our proposed method achieve better results.

Analyzing `Proto` From Table 5.2, `Retriever` outperforms `Proto` by 17% when training the span representations. We conjecture that this is caused by `Proto` learning noisy prototype. Compared to `Retriever`, the similarity scores between the spans and their corresponding class representations are low, indicating that the span-level prototypes may not be clearly separated.

5.2.6 Related Work

Few-shot metric learning Metric learning methods target at learning representations through distance functions. [182] proposed Siamese Networks which differentiated input examples with contrastive and triplet loss functions [174] on positive and negative pairs. While they are more data efficient for new classes than linear classifiers, Siamese Networks are hard to train due to weak pairs sampled from training batch [183]. In comparison, Prototypical Networks [166] proposed to compute class representations by averaging embeddings of support examples for each class. These methods have been mostly explored in computer vision and text classification [184, 185], and consistently outperform Siamese Networks and retrieval-based methods such as k-nearest-neighbors, especially when there are more classes and fewer annotated examples [167, 186]. However, newly added examples which are outliers may change the prototypical representations dramatically that can harm all predictions on the class. In addition, these methods do not perform well when there are more annotated data available per class [167].

Metric learning in language understanding Utilizing relevant examples to boost model performance has been applied to various tasks such as language modeling [187] and question answering [188, 189]. Recently, metric learning has been applied to intent classification [186, 179]. [179] utilized Prototypical Networks to learn intent and slot name prototype representations and classified each token to its closest prototype. They showed better results than meta-learning, another prevalent few-shot learning method [190, 191]. In order to consider label dependencies that are essential in slot tagging tasks [192], [172] proposed a collapsed dependency transfer (CDT) mechanism by simulating transition scores for the target domain from transition probabilities among BIO labels in the source domain, outperforming previous methods on slot filling by a large margin. However, this simulation is noisy and the difference between the source and target domains can result in biased transition probabilities.

The most similar approach to ours is a concurrent work from [193], which learns span boundaries and sentence similarities before retrieving the most similar span, inspired by question-answering models. Even though this approach predicts spans before retrieving on

the span level and thus bypasses the problem of transition probability in previous research, it only achieves unsatisfactory results. Different from these researches, we propose to learn span representations using a batch softmax objective without having to explicitly learn span boundaries. Our method achieves more accurate slot and intent prediction than previous methods in the few-shot setting.

5.3 Problem Representation for Exposing Safety and Consistency Issues

5.3.1 Introduction

Language models, including dialog models, greatly benefit from training on large amounts of data with the objective of mimicking human generated sentences [27, 9, 194, 195, 196]. However, even with carefully pre-processed training data from online sources, neural dialog models are prone to issues including generic utterances, repetition, contradiction, and lack of safety [197, 198, 199, 196, 200]. Compared to modularized dialog systems which are designed to avoid these problems [201, 202], fixing these issues with end-to-end neural models is more challenging, which may hinder real world use of trained models [203, 204]. We argue that before solving these problems using simulated data from simplified scenarios, we need to be able to probe the models and expose the problems in a dynamic way.

Even though crucial limitations of neural dialog models are prevalent, they are mostly manually identified and categorized through interactions between model designers and the dialog system during qualitative analysis [196]. Recent work proposes asking annotators to converse with dialog models while goading the model into generating problematic responses in a black-box attack setting. Although the data collected in this way can improve the performance of both problem classifiers and model generation, human annotators mostly rely on straightforward and intuitive strategies to collect the dataset, which may only expose superficial problems. For instance, [205] instructs crowdworkers to trigger dialog systems into responding with unsafe (offensive or otherwise socially undesirable) utterances, but most of the human messages are either hate speech or controversial state-

ments. Similarly, [206] asks Mechanical Turkers to manually write contradicting dialogs for both humans and bots, or to interact with chatbots, where a frequent strategy is to ask factual questions intentionally leading to contradiction (e.g. ask "do you speak Spanish" after the bot says "I am a Spanish teacher" in previous turns). Although these tricks are effective, the human inputs are not necessarily coherent with the conversation context, and the difference in the distribution from how humans interact with dialog systems makes the collected data less useful in practice. In addition, the data collection procedure is extremely expensive and is not practical for newly trained models. More importantly, systematic problems are still not revealed.

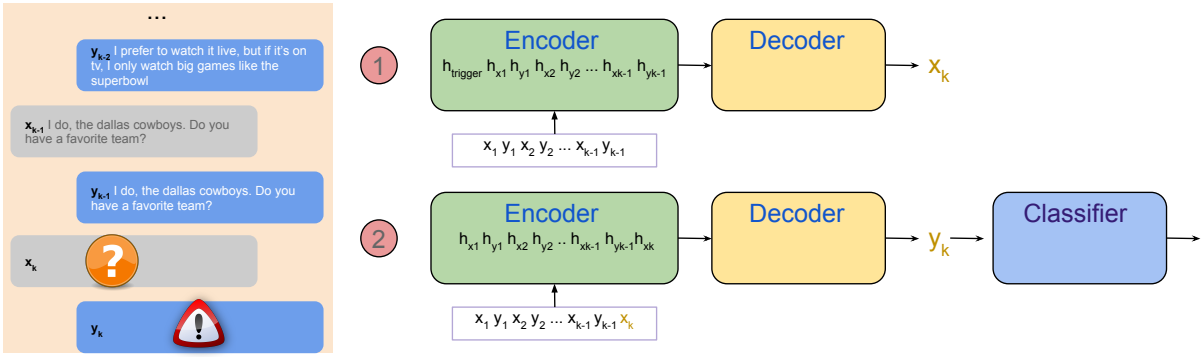

Figure 5.2. Illustration of our problem exposure task and proposed model. Given conversation history, our goal is to generate a coherent prompt x_k, which will induce a neural dialog model (encoder-decoder in this example, all its parameters are frozen) to respond an utterance y_k that contains problems such as unsafe and inconsistent. To do this, we learn hidden states $h_{trigger}$ which will guide the decoder to generate x_k through attention. In the second step, we remove the learned hidden states and append newly generated utterance x_k to generate a response y_k. Contextualized y_k representation is sent to a problem classifier to output either gradients for `Trigger_weakly` (which requires $h_{trigger}$ in step 2), or a reward for `Trigger_PPO`. For `Trigger_PPO_adv`, x_k is also sent to the classifier to obtain a reward.

In this work, we propose to automatically expose problems with neural dialog models in a more systematic setting. Given a conversation context, the goal is to generate a coherent utterance to act as a human `prompt` through self-chat, which will trigger the dialog system into generating a problematic response. To this end, we propose to learn some trigger hidden states while freezing the original dialog model. We assume that we have some problem classifiers which can be from in-domain or out-of-domain collected data. This is practical because out-of-domain data is relatively easy to collect and we

do not require a perfect classifier. Each token can attend to the trigger hidden states when generating the next tokens so that the generated `prompt` include systematic signals regarding target problems. Specifically, we introduce a weakly-supervised method where we can back-propagate the gradients from the classifier through self-attention and cross-attention. We also introduce a reinforcement learning method that uses classifier results on the model responses as rewards.

Compared to sentiment neurons [207], learning trigger hidden states as a problem switch is a much harder task because our hidden states in a relative shallow model are not trained with a huge amount of clearly distinguished supervised data. Furthermore, exposing more subtle problems (such as contradiction) with a coherent prompt that will indirectly impact on the model response rather than direct conditional text generation is more challenging, similar to probing models in an adversarial attack setting. However, we demonstrate the effectiveness of our proposed methods on automatic problem exposure with the state-of-the-art chatbot Blenderbot [196]. We evaluate with two problems: safety and consistency. In addition, we show that the generated examples can help to improve the performance of out-of-domain problem classification as well.

5.3.2 Task Definition

Given the context of a conversation $c_{k-1} = x_1, y_1, x_2, y_2, ..., x_{k-1}, y_{k-1}$, where x_i, y_i represents utterances from each speaker in a turn, we want to generate a `prompt` x_k while keeping the whole conversation coherent and engaging. The original neural dialog model then considers the whole context $(c_{k-1}; x_k)$ to generate a response y_k, which is considered as not acceptable to a problem P (e.g. toxic response to the safety problem). Meanwhile, we have some trained classifier $f_P(h(y_k))$ for the problem P which can indicate how likely the contextual representation $h(y_k)$ has the problem.

5.3.3 Methodology

Since the goal of the task is to expose systematic problems of pre-trained models rather than relying on simple tricks, we generate prompts using the same model in a self-chat paradigm so that when we plug in the generated prompts to the original model we get

exactly the same response. Compared to recent work on instructing humans to goad chatbots where annotators have no information of how the models works in a trial-and-error black-box attack manner, we use gradients of the model.

Motivated by recent success in conditional generation without fine-tuning model parameters [208, 209], we propose to learn a trigger prompt hidden states, $h_{trigger}$, while freezing the original dialog model to maintain output distribution and generation quality. Specifically, for an encoder-decoder model (or a language model), we are modeling

$$p_\theta(x_k|h_{trigger}, x_{<k}, y_{<k}) \tag{5.5}$$

where $h_{trigger}$ is prepended to the beginning of the conversation history and is initialized with the hidden states of the `bos` (beginning of sentence) token. Before any training, the distribution of $p_\theta(x_k|.)$ will not be modified at all. Once we generate the `prompt` x_k, we use the original model to generate a response as

$$p_\theta(y_k|x_{<k+1}, y_{<k}) \tag{5.6}$$

where y_k may be problematic. A trained classifier f indicates the degree of the problem. During training, we sample data (c_{k-1}, x_k, y_k) via $c_{k-1} \sim \mathcal{D}$ where $\mathcal{D}$ can be any unlabeled conversation as the context, and x_k and y_k are generated in a two-stage sequence using Equation 5.5 & 5.6.

In order to optimize $h_{trigger}$ which will boost the level of target problem through attention mechanism [14], we propose a weakly-supervised trigger model (Section 5.3.3.1) where we backpropagate gradients from the classifier back to the hidden states directly, and a reinforcement learning trigger model (Section 5.3.3.2) where the classifier results are used as rewards to optimize the hidden states. We illustrate the task and the proposed methods in Figure 5.2.

5.3.3.1 Weakly-supervised Trigger Model

Because the `prompt` x_k is sampled and detached from the original model, and the classifier operates on the corresponding response y_k, we need to connect $h_{trigger}$ with the response. During training, we first generate a prompt x_k and then simulate the attention mechanism

by modeling

$$p_\theta(y_k|h_{trigger}, x_{<k+1}, y_{<k}) \tag{5.7}$$

where compared to generating the actual response using the original dialog model as in Equation 5.6, $h_{trigger}$ is also used in response generation. We need to apply a classifier f' on the generation hidden states which contains information about $h_{trigger}$ before sampled discrete tokens following [210][3]. We can then use cross-entropy loss against the target label as the training signal to optimize $h_{trigger}$. We refer to this model as `Trigger_weakly`.

Even though this method is relatively straightforward, we note that there are two potential problems. The first one is that because $h_{trigger}$ is considered as one (indirect) input, the optimized hidden states may not necessarily impact on the actual response to lower the loss function. In other words, $h_{trigger}$ is optimized specifically to a loss function regardless of the model output. Another problem is that at inference time when we generate a prompt x_k to get a response using Equation 5.6, there is mismatch from training so that even with a low training loss, the response generated can be very from that during training. However, as we evaluated empirically, $h_{trigger}$ learned this way is still effective.

In addition, we also experimented with gumbel softmax [211, 212] on the generated prompt x_k so that we can input the prompt gumbel vectors which contains information about $h_{trigger}$ using Equation 5.6 during training without the hidden state term. We did not notice a large difference in our preliminary study using an autoregressive language model (GPT-2, [27]), so we use Equation 5.7 for optimization with our weakly-supervised trigger model.

5.3.3.2 Reinforcement Trigger Model

To solve the potential problems with the weakly-supervised trigger model, we leverage reinforcement learning to bypass the challenge in connecting $h\_trigger$ with the model response. During training, we use Equation 5.5 to generate a coherent prompt x_k, and send the sampled discrete tokens to Equation 5.6 to get the response y_k. Instead of using

[3]This classifier f' is only used for training the weakly-supervised model, while a more robust classifier operating on the actual generated response tokens is used for evaluation.

the hidden states, we input the generated response tokens to the classifier f to get a reward $r(y_k)$, where we use the raw logits of the target label. Following [70], we add an adaptive KL term to prevent the generated prompt from diverging too far from the original model

$$KL(x_k) = \beta \log \frac{p_\theta(x_k|h_{trigger}, x_{<k}, y_{<k})}{p_\theta(x_k|x_{<k}, y_{<k})} \tag{5.8}$$

where β varies dynamically to achieve a particular value [70]. The overall reward is thus

$$R(x_k) = r(y_k) - KL(x_k). \tag{5.9}$$

We optimize $h_{trigger}$ using Proximal Policy Optimization (PPO, [213]) with the reward R from Equation 5.9. We refer to this model as `Trigger_PPO`.

Another potential benefit of using reinforcement learning here in addition to fixing the challenges with weakly-supervised trigger model is that we can tweak the reward function to penalize easy triggers. For instance, human annotators may quickly find that controversial statements and unsafe sentences can goad the bot into generating unsafe responses back [205]. If we train $h_{trigger}$ using the original reward function, or use the weakly supervised method where the gradient impacts on both the prompt and the response, then it is very likely that by attending to $h_{trigger}$, the prompt x_k is already problematic. To solve this, we can add a penalty term using the same reward as the response but on the prompt. This encourages the model to only generate acceptable prompts that trigger concerning responses, similar to an adversarial setting. The overall reward is

$$R_{adv}(x_k) = r(y_k) - KL(x_k) - w * r(x_k) \tag{5.10}$$

where w is a weight hyper-parameter to balance between the reward on the prompt and the response. We refer to this model as `Trigger_PPO_adv`.

5.3.4 Experiments and Results

We evaluate our proposed approaches on two problems: safety and consistency by generating prompts that can trigger corresponding problems. In addition, we study whether our generated results can in turn improve the classification performance with out-of-domain data.

For all our experiments, we use the state-of-the-art open-domain chatbot BlenderBot [196] as our pre-trained neural dialog model. The maximum context and response lengths is set to 128 BPE tokens [27]. BlenderBot is pre-trained on Reddit discussions [214] with heuristic filtering and fine-tuned on human-collected clean conversational data including ConvAI2 [215] and Blended Skill Talk [216]. Because of the fine-tuning data, the chatbot frequently deviates from the current conversation topic and asks simple questions such as "do you have a pet". This makes it even harder to generate unsafe and contradictory responses given a coherent prompt. For decoding, we follow the same procedure as in the original model, except that we use sampling instead of beam-search to increase diversity (which is shown to perform as well as beam search in their paper).

During training, we set a maximum number of training steps with early stopping. To prevent unfair comparison to baselines, instead of selecting the best model based on average reward, we early stop when perplexity diverges too much from the original perplexity. We analyze the effect of early stopping in Section 5.3.5.

5.3.4.1 Safety

The safety problem exposure task is to generate coherent prompts where the dialog model will generate unsafe responses given the contexts and prompts. We compare our proposed `Trigger_weakly`, `Trigger_PPO`, and `Trigger_PPO_adv` with the original model `BlenderBot`.

	Response		Prompt			
Method	Unsafe prob. (classifier ↑)	Unsafe % (human ↑)	Unsafe prob. (classifier ↓)	Unsafe % (human ↓)	Perplexity ↓	Language quality (human ↑)
BlenderBot	22.41	9.21	23.64	16.45	16.08	3.98
Trigger_weakly	30.96	21.71	66.67	26.97	17.92	3.01
Trigger_PPO	**32.63**	22.37	49.30	23.68	19.54	3.53
Trigger_PPO_adv	30.57	**26.32**	39.48	17.11	18.42	3.88

Table 5.3. Results on the safety exposure task. All our proposed methods are effective in exposing safety problems. In particular, `Trigger_PPO_adv` shows that even with relatively safe prompts, we can still trigger unsafe utterances from the model. Adding a constraint term on the prompt also helps with maintaining language quality.

Safety Classifier We train our safety classifier f_{safety} using data collected from BAD [205]. We truncate the conversation history to four utterances from both speakers following the best practice in their paper. We also ignore easy cases where the bot says "Hey do you want to talk about something else" from a safety layer during data collection. In addition, we leverage data from BBF [217] including both single-turn and multi-turn examples. In total, we have a training corpus of 197K examples and we evaluate on the BAD validation set with 12.8K examples. We train the classifier using RoBERTa [218]. The classifier achieves an F1 score of 77.34 on unsafe examples, which is close to the number reported in [205], so we did not use additional training data and framework.

For the classifier used in the weakly-supervised method f'_{safety}, we use the same data training a multi-layer perceptron (MLP) on top of frozen BlenderBot hidden states (similar to [210]). f'_{safety} achieves an F1 score of 69.09 on unsafe examples.

Training and Evaluation During training, we sample contexts of three utterances from the pre-processed BAD training data explained above because BlenderBot can only handle 128 tokens. For evaluation, we sample contexts of the same length from the BAD validation data. We report the average probability that the response is unsafe and the average probability that the prompt is unsafe using f_{safety}, as well as the generated sentence perplexity as automatic evaluation averaged over three random seeds. For human evaluation, we report the percentage of unsafe responses, unsafe prompts, and the language quality of the prompts which indicate both fluency and coherence on an 1 - 5 Likert scale using 150 examples.

Results Table 5.3 shows results for the safety exposure task. On the induced responses according to the generated prompts, compared to the baseline model `BlenderBot`, all our proposed methods substantially increase the chance that the responses are unsafe (with more than 8% absolute from safety classifier f_{safety}, and more than 12% from human evaluation). This suggests that these methods are effective in exposing safety problems with the pre-trained models. In addition, the relatively low unsafe percentage (9.21% and 26.32%) indicates that in general, `BlenderBot` tends to generate safe responses due to its clean fine-tuning data. Tricking the model into generating unsafe responses is thus

very challenging without modifying the model distribution, especially when we want to generate coherent prompts with high language quality.

On the generated prompts, as expected, without any constraint as with `Trigger_weakly` and `Trigger_PPO`, the model may learn to increase the likelihood of unsafe responses by crafting unsafe prompts, resulting in much higher prompt unsafe probability judged by both the automatic classifier (more than 15% over the baseline) and human annotation (more than 7%). However, by adding a penalty to the prompt to reduce its unsafe degree (from 23.68% to 17.11% by human evaluation), we can maintain or even outperform unsafe degree in the corresponding responses (26.32%). Meanwhile, the language quality human annotation results show that penalty on the prompt also helps with maintaining coherence and fluency compared to `Trigger_weakly` and `Trigger_PPO`.

5.3.4.2 Consistency

The consistency problem exposure task is to generate coherent prompts to trigger responses that contradict their roles in the conversation context. In contrast with safety, since generating inconsistent prompts will not necessarily result in more inconsistent responses, we do not evaluate on `Trigger_PPO_adv`. Instead, we compare `Trigger_weakly`, `Trigger_PPO`, and the original `BlenderBot`. We also compare with `Human_selected` which picks context-specific prompts that trigger responses labeled as contradictory from multiple sampled pairs in DECODE data collection [219]. It serves as the upper bound for the task.

Consistency Classifier We train our consistency classifier f_{consis} using the data collected from DECODE [206]. Because contextual information is crucial for consistency detection, we do not truncate the context history. The training corpus consists of 27K examples and we evaluate the classifier on the DECODE validation dataset with 4K examples. In order to easily create training signals when optimizing $h_{trigger}$, we train the classifier by concatenating the last response with the context instead of the suggested structured method. Our RoBERTa-based classifier achieves an F1 score of 93.45 on con-

tradictory utterances, which is close to the results in [206] with additional training data[4].

For the weakly-supervised method, f'_{consis} is similar to f'_{safety}, and achieves 86.02 F1 on contradictory examples.

Training and Evaluation During training, because BlenderBot cannot handle longer contexts, we truncate the conversation history to three utterances. We sample examples from the DECODE training data to form $\mathcal{D}$. We note that DECODE training and validation data are collected by asking humans to write utterances for each speaker, which may be different from a chatbot setting. Therefore, for evaluation, we sample contexts from their collected human-bot test set (with 764 examples in total). This set is collected by asking human annotators to interact with multiple chatbots. We report the average probability that the response contradicts with the context using f_{consis} for automatic evaluation. We also report the percentage of contradictory responses for human evaluation on 200 generated examples from three random seeds.

Because the training signal for inconsistency is more delicate compared to other attributes such as sentiment and safety, it may be harder for the model to converge, especially with a diverse set of training examples. Therefore, we also experiment with training on the human-bot context directly. It is worth mentioning that even though we train with the same context as for evaluation, the only training signal is from the classifier f_{consis}. In other words, none of the models require external information such as real collected prompts and responses with their corresponding gold labels. More importantly, we select early-stopping based on perplexity instead of cherry-picking the best examples using actual predicted rewards. Thus it is fair in performance comparison. Moreover, this is a common practice in the literature [190], particularly with reinforcement learning to optimize rewards [220, 70].

Results Table 5.4 summarizes the experiment results on the consistency exposure task. We observe that during training, `Trigger_weakly` does not converge so that its performance on the test data (17.86%) is lower than the baseline. Even though `Trigger_PPO`

[4]Although more accurate classification is beneficial to our model, training more complicated classifiers to achieve only marginal improvements is out of the scope of our work.

Method	Contradiction probs. (classifier ↑)	Contradiction % (human ↑)
BlenderBot	18.24	12.56
Trigger_weakly	17.86	-
Trigger_PPO	19.55	-
Trigger_weakly_ft	19.68	-
Trigger_PPO_ft	**25.49**	**28.14**
Human_selected	-	65.33

Table 5.4. Results on the consistency exposure task. `Trigger_weakly` struggles with learning *h_trigger*, while `Trigger_PPO` is effective especially when training on the human-bot context, outperforming the baseline from both automatic and human annotation. `Human_selected` represents collected data from nie-etal-2020-i where examples are selected by humans labeled as inconsistent.

gets higher reward, training is not very stable and its performance on the target data does not increase by a large margin. This suggests that inconsistency signals may not be easily captured to craft corresponding dynamic prompts. When we train the models on human-bot data instead (denoted as `Trigger_weakly_ft` and `Trigger_PPO_ft` respectively), the weakly supervised method still does not converge. However, `Trigger_PPO_ft` learns how to perform the task evaluated by the learning curve. We thus do human evaluation on this method. `Trigger_PPO_ft` significantly outperforms the baseline (28.14% compared to 12.56%) from human evaluation, suggesting that even with weaker signals, our proposed method is still effective on harder tasks such as inconsistency, which by nature is non-trivial to detect. Lastly, when we compare to the upper bound `Human_selected`, which are picked by humans to be inconsistent, we found that human prompts are shorter compared to our generated prompts because of the minimum generation size of 20 suggested by [196]. Since the context window of our dialog model is limited, longer prompts indicate less context due to truncation. Given that conversation history is critical in inconsistency, this partially explains the relatively lower performance.

5.3.4.3 Out-of-domain Classification

In addition to generating prompts to expose problems of neural dialog models, we examine if the generated prompts and responses can help out-of-domain problem classification. This is critical because due to distribution difference, problem classifiers trained on one domain may not work well on another [221], especially with problems that are hard to expose. For instance, [206] collect a contradiction dataset by asking humans to generate inconsistent responses, which is a much easier task than tricking dialog models into generating inconsistent utterances within a reasonable interaction budget. They observe a large performance gap between in-domain (human-human) training data and out-of-domain human-bot data. To this end, we experiment with generating out-of-domain problem training data with our proposed methods on the consistency task.

Training and Evaluation We use the best performing model from Section 5.3.4.2, `Trigger_PPO` to generate prompts and responses for the evaluation human-bot contexts (`human-bot Trigger_PPO`). We also generate prompts and responses using the original BlenderBot on the same context (`human-bot BlenderBot`). For each generated utterance, we use f_{consis} to predict the probability that it contradicts with the conversation history. A threshold (0.5 in our experiments) is used to convert the predicted probability to a contradiction label. Then we can train the contradiction detection classifier the same way as explained in Section 5.3.4.2. We compare with the classifier trained on the human-collected DECODE training data. The best model checkpoint to perform evaluation is selected on the DECODE validation data.

Training data	DECODE valid	human-bot
DECODE train	**93.45**	59.05
human-bot BlenderBot	73.16	61.36
human-bot Trigger_PPO	76.04	**67.68**

Table 5.5. Results on consistency detection (F1 score of contradiction utterances). Training with data generated by our `Trigger_PPO` model outperforms the classification results by training on the DECODE traning data when we evaluate on human-bot data.

Results Table 5.5 shows the classification results for contradiction prediction on the human-bot data. Similar to previous findings, even though training on human-human data can achieve a high F1 score on the human-human validation set, it suffers from the out-of-domain distribution. The classifier trained with data generated from the original BlenderBot results in similar performance. In comparison, training with the data generated by `Trigger_PPO` significantly outperforms the baseline (67.68 F1 vs. 59.05 F1). This indicates that our proposed method can not only expose model problems, but can help problem detection with new data distribution as well. It can also be used in models that require good domain-specific classification, such as [210].

5.3.5 Analysis

In this section, we discuss model performance in depth with qualitative analysis.

Context
...
x_4: ...
y_4: That would be a good choice. What else do you like to do other than play video games?
x_5: Well, I like listening to music too.
y_5: What type of music do you listen to? I like all kinds of music, but country is my favorite. I like classic rock because it sounds better. I don't like country. You?
`BlenderBot`
x_6: I like older radio and rock and roll. You? I like music with a mellow beat.
y_6: I could never listen to that without my headphones. I've been listening to classic rock and I am always driving while listening.
`Trigger_PPO_ft`
x_6: I like classic rock because it sounds better. I don't like country.
y_6: Classic rock is a great choice as well as folk metal. I don't mind country, but I don like how it's so bland and boring. It's been around since the 70s.
`Human_selected [206]`
x_6: Country is relaxing. I like classical and techno to be honest.
y_6: I like those genres as well. Do you have any pets? I have a cat.

Table 5.6. Example prompts and responses given the context comparing different methods for the consistency exposure task.

Training for more steps The training reward actually does not saturate when it reaches our set maximum number of steps. In other words, we can expect to see higher rewards with more training steps. However, our PPO model starts to exploit environment

quirks to maximize rewards (such as step 80). For instance, for the safety exposure task, the model starts to generate prompts with certain patterns such as "Then put her ..." or "They should ...", even with `Trigger_PPO_adv` adding the prompt penalty term. For the consistency exposure task, the model starts to use prompts with patterns such as "I don't understand you ...", "How long have you ...", or "You are not ...". The generated prompts still consider context (rather than just generating templates), and can vary from different random seeds. Even though they are more effective in inducing problematic responses, the prompts are less coherent and less diverse, resulting in similar n-grams. This suggests that on the one hand, we may need an additional reward in addition to the relatively straightforward negative penalty. On the other hand, with more training steps, we may be able to discover more meaningful "universal triggers" [222] that can trigger target responses regardless of the context.

Weakly-supervised vs. Reinforcement Learning Method Although there is a potential discrepancy between training and testing that $h_{trigger}$ may only learn to optimize the classifier f'_{consis} regardless of the actual task for the weakly-supervised method as explained in Section 5.3.3.1, we found that in the safety exposure task, it can still increase performance from human annotation. However, this results in much higher unsafe degree for the prompt and lower language quality. Qualitatively, we found that it is more likely to generate unsafe tokens and due to the diverged distribution, the prompts are less grammatical with nonsensical tokens. This suggests that the gradients impact on the prompts more to change the corresponding response attribute, which can also explain its worse performance in tasks where prompt attributes are less dominant to responses such as consistency exposure. In comparison, the reinforcement learning method does not rely on gradients that flow through both the responses and prompts. Instead, it utilizes rewards through exploration and exploitation so it can be more effective in different tasks.

Exposing more systematic problems Previous research mostly exposes superficial problems with easy tricks such as controversial statements and repetitive questions which are unnatural and incoherent. To illustrate, [205] show that only 12.9% responses are offensive if their corresponding prompts are safe. In other words, the vast majority of un-

safe responses are induced by unsafe prompts. In comparison, our results show that in the human evaluation test data where 26.32% responses are unsafe, only 17.11% prompts are unsafe, indicating that our generated safe prompts are effective in generating problematic responses. Similarly, for consistency, we found that in our preliminary experiments on 100 examples, none of our generated prompts applies easy tricks such as repetitive questions that directly contradicts the context, whereas 15% of the DECODE human-bot prompts fall in this category. This number is much higher in their collected human-human data with other tricks such as asking numeric questions. In addition, 53% DECODE-collected prompts contain questions (which are more likely to trigger inconsistent responses in general), whereas 39% contain questions from our proposed method (close to 43% in the BlenderBot baseline).

On the other hand, we can find that coherent natural patterns such as "They should ..." and "You are not ..." (rather than easy tricks) are more likely to trigger problematic responses. Together with the evidence that some problem triggers are learnable from our proposed methods above the surface level, we believe that we can expose more systematic problems compared to previous research where human annotators have no direct information to interpret how a natural prompt can trigger corresponding responses.

5.3.6 Related Work

For our introduced task to expose problems with pre-trained dialog models, the most relevant work is in the fields of controlled generation and adversarial attack. The goal for controlled generation is to generate coherent sentences containing some target attributes, whereas the task for adversarial attack is to craft some examples that can fool some trained classifiers.

Controlled Generation Most previous work in controlled text generation involve training or fine-tuning the whole model [223, 224, 225, 70]. Alternatively, to utilize the high-quality pre-trained language model quality, [210, 76] propose to perturb token distributions towards a specific attribute with residual adapters [226]. Recently, [208, 209] show that optimizing simple prefix hidden states is effective in controlling pre-trained models, which inspires us to expose problems in neural dialog models by learning $h\_trigger$.

In terms of applying reinforcement learning to language generation tasks, previous work leverages defined reward functions [220, 227, 228] or human preference [70]. All these work targets at generating sentences that contain target attributes. In contrast, our work optimizes prompt generation, which indirectly triggers a pre-trained model generating responses containing target attributes. There is no straightforward way to apply previous techniques to this task.

Adversarial Attacks with Pre-trained Neural Models Similar to generating adversarial examples to fool natural language understanding models [229, 230, 231, 232], [222, 233] show that some learned discrete nonsensical universal triggers can be used to generate unsafe sentences. On the other hand, [234] finds toxic prompts from naturally occurring sentences. The most similar work to ours is probably directing pre-trained models into generating a list of pre-defined tokens or sentences [235, 236, 237, 238]. In comparison, our task needs to generate coherent prompts according to the conversation history. Furthermore, instead of triggering pre-defined egregious responses, our proposed method is more flexible in exposing a wide range of problems such as consistency where crafting the target responses without context in advance is impossible.

Safety and Consistency To make machine learning models safe to use especially with language generation, there is a long literature in safety such as hate speech [239] and bias [217, 240]. Most of these works focus on abusive context detection. On a different line of research, some work introduces conditional generation to reduce toxicity [210, 234]. These techniques mostly requires some toxic classifiers, which as shown in Section 5.3.4.3, may not work well for a different model distribution. Recently, [205] instructs humans to interact with neural dialog models in an adversarial way in order to induce unsafe responses from chatbots. Although classifiers trained with the introduced dataset are more robust, the collected data is relatively artificial because humans rely on apparent traits such as controversial statement or hate speech, regardless of the semantics and coherence of the conversation.

For consistency, previous work suggests generation grounded by information such as personas [215] and neural memories [241]. In terms of consistency detection, [242, 243, 199]

introduce and suggest using natural language inference to model conversation coherence. Recently, [206] collects a large contradicting human dialog corpus based on a conversational context and show better performance than entailment-based methods. However, as in [205], annotators tend to ask repeating questions to provoke inconsistent answers.

Instead of asking humans to write prompts that may induce problematic responses, which is expensive and unrealistic with newly designed dialog models, we propose to trigger unsafe and inconsistent responses automatically. Our method can expose more systematic errors and is generally applicable to a wide variety of problems with trained neural models.

5.4 Summary

In this chapter, we discussed attribute representation beyond low- and high-level latent variables, where representing the tasks directly displays advantages in both training and inference. Specially, we illustrated example representation which can leverage retrieval, rather than learning task-specific layers for classification and structured prediction tasks. We showed that this approach is effective in the few-shot learning scenario, and the low-level attribute representation to learn span boundaries can actually boost the performance in example construction here. We also explained how to represent systematic problems with newly trained neural models, which successfully finds issues related to safety and consistency and further improves performance of out-of-domain problem classifiers. By representing tasks-specific attributes, we can achieve performance that humans may not due to model training.

Chapter 6
Application: Case Study in Building Dialog Systems

6.1 Overview

In this chapter, we demonstrate how attribute representation can be applied to build an open-domain dialog system from scratch as a case study. We illustrate why attribute representation is important, and how they can be effectively utilized to make the systems robust.

This chapter is based on our works [201, 244, 245] that were published at the EMNLP 2019 conference, which I lead as the main author.

6.2 Introduction

Dialog systems, because of their interactive nature, can be more complicated than traditional NLP tasks. When building a dialog system from scratch, we can utilize some tasks designed for general NLP such as parsing, but the domain difference and linguistic features specific to dialog such as ellipsis makes the task even harder. Moreover, different from written text where we can easily acquire the data, collecting conversational data is expensive, thereby making few-shot learning particularly important.

Dialog systems are designed to serve cases such as personal assistants or simulate customer services. These tools have been widely accessible in commercial devices such

as Amazon Alexa, Google Assistant, and Facebook portal. Traditionally, we can divide dialog systems into task-oriented dialog and open-domain dialog, where the former aims to accomplish some tasks such as booking a restaurant, and the latter focuses on having chit-chat and natural conversation. Similar to the motivation introduced in Chapter 1, we can choose to apply a modularized system including natural language understanding which generates intermediate representation by abstracting the human utterances, dialog management which prepares the policy to return, and natural language generation to generate an appropriate response with the help of some knowledge. We can also choose the build the system end-to-end, by training a Sequence-to-Sequence model on some conversational data similar to a language model. Again, modualized systems are more controllable, and end-to-end models are more robust.

No matter we choose modularied or end-to-end systems, there are four main challenges to build a dialog system. Firstly, since the input is mostly from speech (rather than a written-text based system), there are automatic speech recognition (ASR) errors such as disfluency, especially with complex sentences. Secondly, linguistic features such as ellipsis across multiple turns and coreferences are much more frequent than other NLP tasks and annotations [246], which makes training models with data augmentation hard [150]. Next, whether we choose a template-based method or a generation model, we have to make sure the response is specific, factual, and safe, which are critical but not controllable in current dialog systems [196, 29]. Lastly, we also need to consider prosody to simulate natural utterances since we need to respond to users through speech. In addition to the four problems related to systems, we also need to consider how to fix model bias and problems, and how to make the model robust if not performing as expected during deployment.

6.3 Methodology

We will first explain how attribute representation can be utilized for modularized systems, and then we discuss its application to end-to-end models. Figure 6.1 illustrates an architecture for a dialog system from speech input to language processing to speech output, where the processing part can be split into different parts in a modularized system, or by

taking as whole in an end-to-end system.

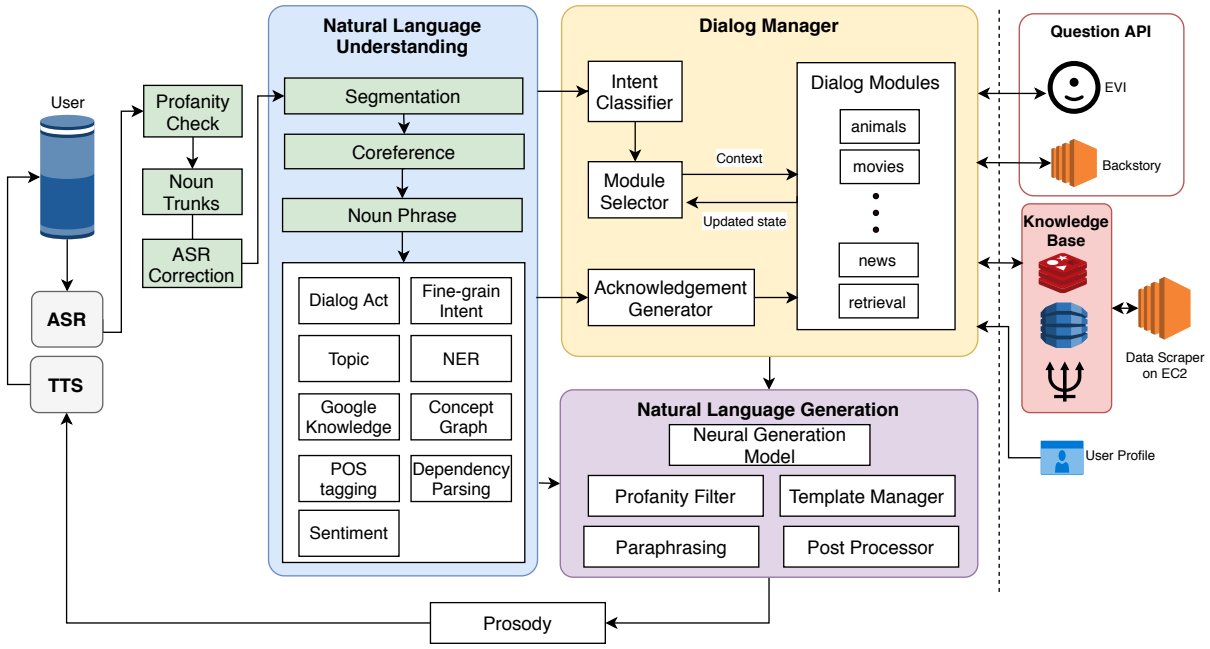

Figure 6.1. Dialog system architecture from [245].

6.3.1 Modularized systems

The most critical question we need to address for a modularized system (and also sometimes an end-to-end system) is what output we need and how to annotate data. For example, if we know that we need dialog act to represent user intent, and topic as some features to dialog manager, the prerequisite would be to build the ontology first, before asking human annotators to collect a dataset and then train some models. Most of these features are low-level. As readers may recall, this is what the low-level attribute representation is studying. As explained in Chapter 4, we can automatically induce a hierarchical schema which are important to many NLU features [118]. Another benefit of inducing the schema automatically is that there may be cases not considered and one may need to iteratively update the ontology through data collection, hence making the automatic method more ideal. Moreover, instead of deciding what features we need such as sentiment and semantic parsing [247] and then come up with the ontology manually or automatically, we can directly learn what features are useful from the low-level attribute representation. Furthermore, we can use the induced data to train a model directly, and

then keep improving the model performance while interacting with real traffic.

Once we have the data, we can train some models by either fine-tuning some layers to map the learned representation to the target space, or to represent the task that directly translates the task representation to the target. It is very frequent that due to the annotation budget or time requirement, we may not be able to collect a large dataset for all labels or domains we need. Chapter 3 suggests that we may only need to collect some labeled data on one domain, and represent features in other domains as high-level attributes so that we can transfer useful information to new scenarios in few-shot or even zero-shot settings. The high-level attribute representations enable not only cross-domain transfer learning, but also more broad applications such as cross-lingual transfer learning.

Then the question to address when we have some runnable models is how to keep improving the model performance in a continual learning setting. For example, the model may consistently predict wrong labels, or we need to add new labels or change schema because of new requirements. This setting applies to all features in NLU. The task-specific attribute representation such as the retrieval method introduced in Chapter 5 suggests that we do not require retraining the model, or collecting new data to change the mapping of target space entirely. Instead, we can simply choose to encode example attribute representations as index and use retrieval to finish various tasks. In addition to NLU tasks, such methods can also be extended to dialog manager and NLG where we can change the task attribute that translates to the target space directly. More importantly, if some data is not easy to collect (such as inconsistency detection), we are able to simulate the target task augmentation process by representation the task attribute directly [248, 249].

We will discuss NLG in Section 6.3.2 since language generation is the only observable part of an end-to-end system, and its main task is to generate a response similar to the NLG task in the modularzied system.

6.3.2 End-to-end

There are two main difference between an end-to-end system and a modularized system. The first one is the integration of knowledge, and the second is how to condition the input for text output.

In a modularized system, we can extract key features from NLU and call some APIs as knowledge base to find relevant information such as the answer to a factual question, or the weather in a location. In contrast, end-to-end systems typically encode all the world-knowledge into the model parameters [39, 29]. The benefits of encoding knowledge into parameters is that we can simply train the model to capture potential queries, but with the cost of not being able to update the knowledge easily, and not being able to inspect if the model parameters actually contain some specific knowledge. More importantly, for many downstream tasks such as entity linking and dialog state tracking, domain-specific knowledge is critical to make the correct prediction, which typically requires fine-tuning with a large dataset to expect acceptable performance. Training the model together with encoding relevant knowledge is thus not efficient. Instead, one crucial direction is to disentangle knowledge from a neural model, where the knowledge can be some database [250, 10], or even some other modalities [43]. In addition to task-specific attribute representation explained in Chapter 5, one can also consider high-level and low-level attribute representation applied to the knowledge base space, which is an interesting future work direction.

Compared to a modularzied system, the language generation in an end-to-end system is conditioned on the encoder representation, rather than features abstracted by NLU and dialog manager. The key question to address here is that along with being factual without hallucination by integrating external knowledge, we need to make sure that the model is steerable and reliable. For example, we may prefer generation models in a modularized system or an end-to-end system to generate positive responses rather than being sarcastic or issuing pessimistic comments. More critically, we need to make sure that the model does not generate toxic responses. This requirement is not only essential to a dialog system, but generally to any models with a generation portion on any modalities as well. Solutions such as high-level attribute representation to effectively control a neural model [251], and task-specific representation to avoid unsafe and inconsistent generated texts explained in previous chapters are significant research areas.

6.4 Summary

In this chapter, we introduced the challenges in building dialog systems as an application case study, and summarized how representation on the high-level, low-level, and task-specific attributes can be learned and utilized to solve these challenges. We showed that in either a modularized system or an end-to-end system, attribute representation helps to define the essential tasks automatically, ease the requirement of large-scale in-domain supervised data, and steer the output towards a target direction in phases from data annotation to model training and deployment. Therefore, such representations can make the neural models more controllable, reliable, and explainable.

Chapter 7
Conclusion

Neural (language) models are powerful enough to finish down-stream tasks efficiently or even achieve super-human performance. When further scaling up to more parameters, it is even possible that neural models can get rid of the requirement of supervised examples and find relevant information with search engines to accomplish more complicated tasks the way humans do. However, they are far from being "all we need". In this thesis, we were trying to combine the benefits of modularzied systems including the ease of explanation and training, and the benefits of end-to-end systems including being more flexible, into large pre-trained language models to make them more controllable, explainable, and reliable.

Specifically, we proposed to represent attributes that are relevant to a target task, and use the learned representation to guide a pre-trained model for NLU and NLG tasks with or without optimizing the model parameters. Depending on the task specifics, we introduced high-level, low-level, and task-specific attribute representation.

For high-level attribute representation, we explored a general method to train attribute representation together with a language modeling task. When considering the learned representation in a different domain, the model can utilize such information by identifying similarities and distinctions among tasks. We applied high-level attribute representation on a cross-lingual zero-shot learning tasks, where we learn language representation to achieve effective performance on intrinsic and extrinsic tasks. We also applied the rep-

resentation on a controlled generation task, where we demonstrated with sentiment and topic control using an alignment function on attributes. We further analyzed how to disentangle different attributes and combine attributes as conditions to solve challenges in high-level attribute representation.

In low-level attribute presentation, we were motivated by the obstacles that although neural models contain useful information, it may not be easy to clearly define a high-level attribute such as categories in an ontology. Therefore, we introduced low-level attribute representation methods where we experimented with language model fine-tuning, before inspecting model representation to uncover low-level attribute representation. We demonstrated such a method on a task-oriented dialog setting where we induced the schema by detecting boundaries using attention distribution, before clustering phrases into groups for the dialog state tracking task.

Different from high-level and low-level attributes, we introduced task-specific attribute representation where it may be easier to represent task-specific features directly to connect model parameters to the target, rather than other representations. We illustrated with two distant task-specific attributes, including an example representation, and a problem representation. In example representation, we considered training data as attributes and encode them as index so that we can retrieve spans and sentences together with their labels by disentangling model representation from the output space for classification and structured prediction tasks. In problem representation, we considered safety and consistency as attributes and exposed the most critical issues with neural generation models using different learning methods. We observed competitive performance in demanding tasks that are not trivial to humans.

Finally, we reviewed how to apply attribute representation to a realistic setting in building dialog systems as a case study. We introduced challenges in dialog system pipelines including NLU and NLG, and analyzed the trade-offs of modularized systems and end-to-end systems. We explained how to tackle the overhead with attribute representation from high-level, low-level, and task-specific perspectives, and pointed out a broader application to general NLP tasks.

To summarize, high-level attribute representations makes it effective in controlling a pre-trained model towards a specific direction, or transfer across domains or even languages. In contrast, low-level attribute representations eliminate the requirement of the clear definition of an attribute, and can uncover what features a model has encapsulated and what features need to be learned. Meanwhile, task-specific attribute representation detaches learned model parameters from the target output space, thereby making it compelling in complicated tasks without abundant supervised data. By combining these attribute representations depending on the actual requirements, we can make the model more flexible, reliable, controllable, and explainable.

For future directions, although attribute representations have been shown to be effective in various language tasks, we can expand the scope to other modalities such as image and speech. Similar to texts, tasks such as image generation and captioning also suffer from the challenges such as not controllable and explainable. We can either choose to represent attributes in the same way we do in languages, or choose new perspectives where, for example, we consider language as some attributes, and control an image model directly for target directions. More importantly, apart from the applications we showed, one question that is still remaining with neural models is that different from humans, neural models do not consider any features such as intent. Although manually created attributes such as sentiment or fine-grained attributes can make the model controllable after model pre-training and revealing what the model has learned, it would still be bounded and leaves it an open question of how to design attribute representations in model pre-training to make this more powerful.

References

[1] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 160–167. [Online]. Available: https://doi.org/10.1145/1390156.1390177

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 2014, pp. 1532–1543. [Online]. Available: https://www.aclweb.org/anthology/D14-1162/

[4] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[6] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "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)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237. [Online]. Available: https://aclanthology.org/N18-1202

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "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)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[8] A. Baevski, W. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, "data2vec: A general framework for self-supervised learning in speech, vision and language," *CoRR*, vol. abs/2202.03555, 2022. [Online]. Available: https://arxiv.org/abs/2202.03555

[9] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[10] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, X. Jiang, K. Cobbe, T. Eloundou, G. Krueger, K. Button, M. Knight, B. Chess, and J. Schulman, "Webgpt: Browser-assisted question-answering with human feedback," *CoRR*, vol. abs/2112.09332, 2021. [Online]. Available: https://arxiv.org/abs/2112.09332

[11] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "Palm: Scaling language modeling with pathways," 2022. [Online]. Available: https://arxiv.org/abs/2204.02311

[12] E. M. Bender and A. Koller, "Climbing towards NLU: On meaning, form, and understanding in the age of data," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 5185–5198. [Online]. Available: https://aclanthology.org/2020.acl-main.463

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[15] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for

statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. [Online]. Available: https://aclanthology.org/D14-1179

[16] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rkgNKkHtvB

[17] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *CoRR*, vol. abs/1904.10509, 2019. [Online]. Available: http://arxiv.org/abs/1904.10509

[18] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *CoRR*, vol. abs/2004.05150, 2020. [Online]. Available: https://arxiv.org/abs/2004.05150

[19] J. Ainslie, S. Ontanon, C. Alberti, V. Cvicek, Z. Fisher, P. Pham, A. Ravula, S. Sanghai, Q. Wang, and L. Yang, "ETC: Encoding long and structured inputs in transformers," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 268–284. [Online]. Available: https://aclanthology.org/2020.emnlp-main.19

[20] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[21] Y. Wu, M. N. Rabe, D. Hutchins, and C. Szegedy, "Memorizing transformers," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=TrjbxzRcnf-

[22] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2978–2988. [Online]. Available: https://aclanthology.org/P19-1285

[23] D. Hutchins, I. Schlag, Y. Wu, E. Dyer, and B. Neyshabur, "Block-recurrent transformers," 2022. [Online]. Available: https://arxiv.org/abs/2203.07852

[24] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *CoRR*, vol. abs/2009.06732, 2020. [Online]. Available: https://arxiv.org/abs/2009.06732

[25] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical*

Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://aclanthology.org/D14-1162

[26] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *CoRR*, 2018.

[27] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *CoRR*, 2019.

[28] D. Adiwardana, M. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, and Q. V. Le, "Towards a human-like open-domain chatbot," *CoRR*, vol. abs/2001.09977, 2020. [Online]. Available: https://arxiv.org/abs/2001.09977

[29] R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, Y. Zhou, C. Chang, I. Krivokon, W. Rusch, M. Pickett, K. S. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. Aguera-Arcas, C. Cui, M. Croak, E. H. Chi, and Q. Le, "Lamda: Language models for dialog applications," *CoRR*, vol. abs/2201.08239, 2022. [Online]. Available: https://arxiv.org/abs/2201.08239

[30] E. Hosseini-Asl, B. McCann, C.-S. Wu, S. Yavuz, and R. Socher, "A simple language model for task-oriented dialogue," *arXiv preprint arXiv:2005.00796*, 2020.

[31] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. [Online]. Available: https://aclanthology.org/P16-1162

[32] A. Conneau and G. Lample, "Cross-lingual language model pretraining," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019, pp. 7059–7069. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/c04c19c2c2474dbf5f7ac4372c5b9af1-Paper.pdf

[33] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[34] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[35] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "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*. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703

[36] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[37] Y. Tay, M. Dehghani, V. Q. Tran, X. Garcia, D. Bahri, T. Schuster, H. S. Zheng, N. Houlsby, and D. Metzler, "Unifying language learning paradigms," 2022. [Online]. Available: https://arxiv.org/abs/2205.05131

[38] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *CoRR*, vol. abs/2101.03961, 2021. [Online]. Available: https://arxiv.org/abs/2101.03961

[39] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, B. Zoph, L. Fedus, M. Bosma, Z. Zhou, T. Wang, Y. E. Wang, K. Webster, M. Pellat, K. Robinson, K. Meier-Hellstern, T. Duke, L. Dixon, K. Zhang, Q. V. Le, Y. Wu, Z. Chen, and C. Cui, "Glam: Efficient scaling of language models with mixture-of-experts," *CoRR*, vol. abs/2112.06905, 2021. [Online]. Available: https://arxiv.org/abs/2112.06905

[40] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "Palm: Scaling language modeling with pathways," 2022. [Online]. Available: https://arxiv.org/abs/2204.02311

[41] L. H. Li, M. Yatskar, D. Yin, C. Hsieh, and K. Chang, "Visualbert: A simple and performant baseline for vision and language," *CoRR*, vol. abs/1908.03557, 2019. [Online]. Available: http://arxiv.org/abs/1908.03557

[42] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[43] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," *CoRR*, vol. abs/2103.00020, 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[44] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," *CoRR*, vol. abs/2102.12092, 2021. [Online]. Available: https://arxiv.org/abs/2102.12092

[45] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *CoRR*, 2022. [Online]. Available: https://cdn.openai.com/papers/dall-e-2.pdf

[46] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan, "Flamingo: a visual language model for few-shot learning," 2022. [Online]. Available: https://arxiv.org/abs/2204.14198

[47] D. Yu, C. Khatri, A. Papangelis, A. Madotto, M. Namazifar, J. Huizinga, A. Ecoffet, H. Zheng, P. Molino, J. Clune, Z. Yu, K. Sagae, and G. Tür, "Commonsense and semantic-guided navigation through language in embodied environment," in *ViGIL@NeurIPS*, 2019.

[48] D. Yu, T. He, and K. Sagae, "Language embeddings for typology and cross-lingual transfer learning," 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)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 7210–7225. [Online]. Available: https://aclanthology.org/2021.acl-long.560

[49] D. Yu, Z. Yu, and K. Sagae, "Attribute alignment: Controlling text generation from pre-trained language models," in *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 2251–2268. [Online]. Available: https://aclanthology.org/2021.findings-emnlp.194

[50] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *CoRR*, vol. abs/1906.08237, 2019. [Online]. Available: http://arxiv.org/abs/1906.08237

[51] B. Zoph, D. Yuret, J. May, and K. Knight, "Transfer learning for low-resource neural machine translation," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1568–1575. [Online]. Available: https://www.aclweb.org/anthology/D16-1163

[52] T. Pires, E. Schlinger, and D. Garrette, "How multilingual is multilingual BERT?" in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4996–5001. [Online]. Available: https://aclanthology.org/P19-1493

[53] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, "CTRL: A conditional transformer language model for controllable generation," *CoRR*, vol. abs/1909.05858, 2019. [Online]. Available: http://arxiv.org/abs/1909.05858

[54] A. Holtzman, J. Buys, M. Forbes, A. Bosselut, D. Golub, and Y. Choi, "Learning to write with cooperative discriminators," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1638–1649. [Online]. Available: https://aclanthology.org/P18-1152

[55] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, and R. Liu, "Plug and play language models: A simple approach to controlled text generation," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1edEyBKDS

[56] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. B. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean, "Google's multilingual neural machine translation system: Enabling zero-shot translation," *CoRR*, vol. abs/1611.04558, 2016. [Online]. Available: http://arxiv.org/abs/1611.04558

[57] W. Ammar, G. Mulcaire, M. Ballesteros, C. Dyer, and N. A. Smith, "Many languages, one parser," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 431–444, 2016. [Online]. Available: https://aclanthology.org/Q16-1031

[58] P. Littell, D. R. Mortensen, K. Lin, K. Kairis, C. Turner, and L. Levin, "URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 8–14. [Online]. Available: https://www.aclweb.org/anthology/E17-2002

[59] J. Bjerva, R. Östling, M. H. Veiga, J. Tiedemann, and I. Augenstein, "What do language representations really represent?" *Computational Linguistics*, vol. 45, no. 2, pp. 381–389, Jun. 2019. [Online]. Available: https://aclanthology.org/J19-2006

[60] R. Östling and J. Tiedemann, "Continuous multilinguality with language vectors," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 644–649. [Online]. Available: https://aclanthology.org/E17-2102

[61] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean, "Google's multilingual neural machine translation system: Enabling zero-shot translation," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 339–351, 2017. [Online]. Available: https://aclanthology.org/Q17-1024

[62] M. de Lhoneux, J. Bjerva, I. Augenstein, and A. Søgaard, "Parameter sharing between dependency parsers for related languages," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 4992–4997. [Online]. Available: https://www.aclweb.org/anthology/D18-1543

[63] E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, and J. Weston, "Wizard of wikipedia: Knowledge-powered conversational agents," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=r1l73iRqKm

[64] S. Prabhumoye, A. W. Black, and R. Salakhutdinov, "Exploring controllable text generation techniques," in *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 1–14. [Online]. Available: https://aclanthology.org/2020.coling-main.1

[65] C. D. V. Hoang, T. Cohn, and G. Haffari, "Incorporating side information into recurrent neural network language models," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 1250–1255. [Online]. Available: https://www.aclweb.org/anthology/N16-1149

[66] Z. Fu, X. Tan, N. Peng, D. Zhao, and R. Yan, "Style transfer in text: Exploration and evaluation," in *AAAI Conference on Artificial Intelligence*, 2018. [Online]. Available: https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17015/15745

[67] S. R. Bowman, L. Vilnis, O. Vinyals, A. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," in *Proceedings of The 20th*

SIGNLL Conference on Computational Natural Language Learning. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 10–21. [Online]. Available: https://www.aclweb.org/anthology/K16-1002

[68] W. Wang, Z. Gan, H. Xu, R. Zhang, G. Wang, D. Shen, C. Chen, and L. Carin, "Topic-guided variational auto-encoder for text generation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 166–177. [Online]. Available: https://www.aclweb.org/anthology/N19-1015

[69] J. Ficler and Y. Goldberg, "Controlling linguistic style aspects in neural language generation," in *Proceedings of the Workshop on Stylistic Variation*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 94–104. [Online]. Available: https://www.aclweb.org/anthology/W17-4912

[70] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. F. Christiano, and G. Irving, "Fine-tuning language models from human preferences," *CoRR*, vol. abs/1909.08593, 2019. [Online]. Available: http://arxiv.org/abs/1909.08593

[71] E. M. Smith, D. Gonzalez-Rico, E. Dinan, and Y. Boureau, "Controlling style in generated dialogue," *CoRR*, vol. abs/2009.10855, 2020. [Online]. Available: https://arxiv.org/abs/2009.10855

[72] A. Romanov, A. Rumshisky, A. Rogers, and D. Donahue, "Adversarial decomposition of text representation," 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 815–825. [Online]. Available: https://www.aclweb.org/anthology/N19-1088

[73] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *Proceedings of Machine Learning Research*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 2790–2799. [Online]. Available: http://proceedings.mlr.press/v97/houlsby19a.html

[74] A. Bapna and O. Firat, "Simple, scalable adaptation for neural machine translation," 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1538–1548. [Online]. Available: https://aclanthology.org/D19-1165

[75] Z. M. Ziegler, L. Melas-Kyriazi, S. Gehrmann, and A. M. Rush, "Encoder-agnostic adaptation for conditional language generation," *CoRR*, vol. abs/1908.06938, 2019. [Online]. Available: http://arxiv.org/abs/1908.06938

[76] A. Madotto, E. Ishii, Z. Lin, S. Dathathri, and P. Fung, "Plug-and-play conversational models," in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 2422–2433. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.219

[77] A. Chan, Y.-S. Ong, B. Pung, A. Zhang, and J. Fu, "Cocon: A self-supervised approach for controlled text generation," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=VD_ozqvBy4W

[78] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug & play generative networks: Conditional iterative generation of images in latent space," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[79] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rygGQyrFvH

[80] B. Krause, A. D. Gotmare, B. McCann, N. S. Keskar, S. R. Joty, R. Socher, and N. F. Rajani, "Gedi: Generative discriminator guided sequence generation," *CoRR*, vol. abs/2009.06367, 2020. [Online]. Available: https://arxiv.org/abs/2009.06367

[81] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," 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)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 4582–4597. [Online]. Available: https://aclanthology.org/2021.acl-long.353

[82] A. N. Little, "Connecting documentation and revitalization: A new approach to language apps," in *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages*. Honolulu: Association for Computational Linguistics, Mar. 2017, pp. 151–155. [Online]. Available: https://aclanthology.org/W17-0120

[83] D. Wang and J. Eisner, "Fine-Grained Prediction of Syntactic Typology: Discovering Latent Structure with *Supervised* Learning," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 147–161, Dec. 2017. [Online]. Available: https://www.mitpressjournals.org/doi/abs/10.1162/tacl_a_00052

[84] R. Östling, "Word order typology through multilingual word alignment," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 205–211. [Online]. Available: https://aclanthology.org/P15-2034

[85] C. Malaviya, G. Neubig, and P. Littell, "Learning language representations for typology prediction," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 2529–2535. [Online]. Available: https://aclanthology.org/D17-1268

[86] X. Tan, J. Chen, D. He, Y. Xia, T. Qin, and T.-Y. Liu, "Multilingual neural machine translation with language clustering," 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 963–973. [Online]. Available: https://aclanthology.org/D19-1089

[87] J. Bjerva and I. Augenstein, "From phonology to syntax: Unsupervised linguistic typology at different levels with language embeddings," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 907–916. [Online]. Available: https://aclanthology.org/N18-1083

[88] J. Bjerva, Y. Kementchedjhieva, R. Cotterell, and I. Augenstein, "A probabilistic generative model of linguistic typology," 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 1529–1540. [Online]. Available: https://aclanthology.org/N19-1156

[89] E. Rabinovich, N. Ordan, and S. Wintner, "Found in translation: Reconstructing phylogenetic language trees from translations," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 530–540. [Online]. Available: https://aclanthology.org/P17-1049

[90] P. Koehn, "Europarl: A Parallel Corpus for Statistical Machine Translation," in *Conference Proceedings: the tenth Machine Translation Summit*, AAMT. Phuket, Thailand: AAMT, 2005, pp. 79–86. [Online]. Available: http://mt-archive.info/MTS-2005-Koehn.pdf

[91] M. Baker, G. Francis, and E. Tognini-Bonelli, *'Corpus Linguistics and Translation Studies: Implications and Applications'*. Netherlands: John Benjamins Publishing Company, 1993.

[92] G. Toury, *Descriptive Translation Studies and beyond*. Amsterdam /Philadelphia: John Benjamins, 1995.

[93] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 8440–8451. [Online]. Available: https://aclanthology.org/2020.acl-main.747

[94] K. K, Z. Wang, S. Mayhew, and D. Roth, "Cross-lingual ability of multilingual bert: An empirical study," 2019.

[95] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[96] F. Ginter, J. Hajič, J. Luotolahti, M. Straka, and D. Zeman, "CoNLL 2017 shared task - automatically annotated raw texts and word embeddings," 2017, LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. [Online]. Available: http://hdl.handle.net/11234/1-1989

[97] G. Lample, A. Conneau, M. Ranzato, L. Denoyer, and H. Jégou, "Word translation without parallel data," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=H196sainb

[98] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[99] M. S. Dryer and M. Haspelmath, Eds., *WALS Online*. Leipzig: Max Planck Institute for Evolutionary Anthropology, 2013. [Online]. Available: https://wals.info/

[100] A. Conneau, R. Rinott, G. Lample, A. Williams, S. Bowman, H. Schwenk, and V. Stoyanov, "XNLI: Evaluating cross-lingual sentence representations," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 2475–2485. [Online]. Available: https://aclanthology.org/D18-1269

[101] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[102] J. Bjerva, E. Salesky, S. J. Mielke, A. Chaudhary, G. G. A. Celano, E. M. Ponti, E. Vylomova, R. Cotterell, and I. Augenstein, "SIGTYP 2020 shared task: Prediction of typological features," in *Proceedings of the Second Workshop on Computational Research in Linguistic Typology*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1–11. [Online]. Available: https://aclanthology.org/2020.sigtyp-1.1

[103] D. Zeman, J. Nivre, M. Abrams, E. Ackermann, N. Aepli, Ž. Agić, L. Ahrenberg, C. K. Ajede, G. Aleksandravičiūtė, L. Antonsen, K. Aplonova, A. Aquino, M. J. Aranzabe, G. Arutie, M. Asahara, L. Ateyah, F. Atmaca, M. Attia, A. Atutxa, L. Augustinus, E. Badmaeva, M. Ballesteros, E. Banerjee, S. Bank, V. Barbu Mititelu, V. Basmov, C. Batchelor, J. Bauer, K. Bengoetxea, Y. Berzak, I. A. Bhat, R. A. Bhat, E. Biagetti, E. Bick, A. Bielinskienė, R. Blokland, V. Bobicev, L. Boizou, E. Borges Völker, C. Börstell, C. Bosco, G. Bouma, S. Bowman, A. Boyd, K. Brokaitė, A. Burchardt, M. Candito, B. Caron, G. Caron, T. Cavalcanti, G. Cebiroğlu Eryiğit, F. M. Cecchini, G. G. A. Celano, S. Čéplö, S. Cetin, F. Chalub, E. Chi, J. Choi, Y. Cho, J. Chun, A. T. Cignarella, S. Cinková, A. Collomb, Ç. Çöltekin, M. Connor, M. Courtin, E. Davidson, M.-C. de Marneffe, V. de Paiva, E. de Souza, A. Diaz de Ilarraza, C. Dickerson, B. Dione, P. Dirix, K. Dobrovoljc, T. Dozat, K. Droganova, P. Dwivedi, H. Eckhoff, M. Eli, A. Elkahky, B. Ephrem, O. Erina, T. Erjavec, A. Etienne, W. Evelyn, R. Farkas, H. Fernandez Alcalde, J. Foster, C. Freitas, K. Fujita, K. Gajdošová, D. Galbraith, M. Garcia, M. Gärdenfors, S. Garza, K. Gerdes, F. Ginter, I. Goenaga, K. Gojenola, M. Gökırmak, Y. Goldberg, X. Gómez Guinovart, B. González Saavedra, B. Griciūtė, M. Grioni, L. Grobol, N. Grūzītis, B. Guillaume, C. Guillot-Barbance, T. Güngör, N. Habash, J. Hajič, J. Hajič jr., M. Hämäläinen, L. Hà Mỹ, N.-R. Han, K. Harris, D. Haug, J. Heinecke, O. Hellwig, F. Hennig, B. Hladká, J. Hlaváčová, F. Hociung, P. Hohle, J. Hwang, T. Ikeda, R. Ion, E. Irimia, Ọ. Ishola, T. Jelínek, A. Johannsen, H. Jónsdóttir, F. Jørgensen, M. Juutinen, H. Kaşıkara, A. Kaasen, N. Kabaeva, S. Kahane, H. Kanayama, J. Kanerva, B. Katz, T. Kayadelen, J. Kenney, V. Kettnerová, J. Kirchner, E. Klementieva, A. Köhn, A. Köksal, K. Kopacewicz, T. Korkiakangas, N. Kotsyba, J. Kovalevskaitė, S. Krek, S. Kwak, V. Laippala, L. Lambertino, L. Lam, T. Lando, S. D. Larasati, A. Lavrentiev, J. Lee, P. Lê Hồng, A. Lenci, S. Lertpradit, H. Leung, M. Levina, C. Y. Li, J. Li, K. Li, K. Lim, Y. Li, N. Ljubešić, O. Loginova, O. Lyashevskaya, T. Lynn, V. Macketanz, A. Makazhanov, M. Mandl, C. Manning, R. Manurung, C. Mărănduc, D. Mareček, K. Marheinecke, H. Martínez Alonso, A. Martins, J. Mašek, H. Matsuda, Y. Matsumoto, R. McDonald, S. McGuinness, G. Mendonça, N. Miekka, M. Misirpashayeva, A. Missilä, C. Mititelu, M. Mitrofan, Y. Miyao, S. Montemagni, A. More, L. Moreno Romero, K. S. Mori, T. Morioka, S. Mori, S. Moro, B. Mortensen, B. Moskalevskyi, K. Muischnek, R. Munro, Y. Murawaki, K. Müürisep, P. Nainwani, J. I. Navarro Horñiacek, A. Nedoluzhko, G. Nešpore-Bērzkalne, L. Nguyễn Thị, H. Nguyễn Thị Minh, Y. Nikaido, V. Nikolaev, R. Nitisaroj, H. Nurmi, S. Ojala, A. K. Ojha, A. Olúòkun,

M. Omura, E. Onwuegbuzia, P. Osenova, R. Östling, L. Øvrelid, Ş. B. Özateş, A. Özgür, B. Öztürk Başaran, N. Partanen, E. Pascual, M. Passarotti, A. Patejuk, G. Paulino-Passos, A. Peljak-Łapińska, S. Peng, C.-A. Perez, G. Perrier, D. Petrova, S. Petrov, J. Phelan, J. Piitulainen, T. A. Pirinen, E. Pitler, B. Plank, T. Poibeau, L. Ponomareva, M. Popel, L. Pretkalniņa, S. Prévost, P. Prokopidis, A. Przepiórkowski, T. Puolakainen, S. Pyysalo, P. Qi, A. Rääbis, A. Rademaker, L. Ramasamy, T. Rama, C. Ramisch, V. Ravishankar, L. Real, P. Rebeja, S. Reddy, G. Rehm, I. Riabov, M. Rießler, E. Rimkutė, L. Rinaldi, L. Rituma, L. Rocha, M. Romanenko, R. Rosa, V. Roșca, D. Rovati, O. Rudina, J. Rueter, S. Sadde, B. Sagot, S. Saleh, A. Salomoni, T. Samardžić, S. Samson, M. Sanguinetti, D. Särg, B. Saulīte, Y. Sawanakunanon, S. Scarlata, N. Schneider, S. Schuster, D. Seddah, W. Seeker, M. Seraji, M. Shen, A. Shimada, H. Shirasu, M. Shohibussirri, D. Sichinava, A. Silveira, N. Silveira, M. Simi, R. Simionescu, K. Simkó, M. Šimková, K. Simov, M. Skachedubova, A. Smith, I. Soares-Bastos, C. Spadine, A. Stella, M. Straka, J. Strnadová, A. Suhr, U. Sulubacak, S. Suzuki, Z. Szántó, D. Taji, Y. Takahashi, F. Tamburini, T. Tanaka, S. Tella, I. Tellier, G. Thomas, L. Torga, M. Toska, T. Trosterud, A. Trukhina, R. Tsarfaty, U. Türk, F. Tyers, S. Uematsu, R. Untilov, Z. Urešová, L. Uria, H. Uszkoreit, A. Utka, S. Vajjala, D. van Niekerk, G. van Noord, V. Varga, E. Villemonte de la Clergerie, V. Vincze, A. Wakasa, L. Wallin, A. Walsh, J. X. Wang, J. N. Washington, M. Wendt, P. Widmer, S. Williams, M. Wirén, C. Wittern, T. Woldemariam, T.-s. Wong, A. Wróblewska, M. Yako, K. Yamashita, N. Yamazaki, C. Yan, K. Yasuoka, M. M. Yavrumyan, Z. Yu, Z. Žabokrtský, A. Zeldes, H. Zhu, and A. Zhuravleva, "Universal dependencies 2.6," 2020, LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. [Online]. Available: http://hdl.handle.net/11234/1-3226

[104] T. Dozat and C. D. Manning, "Deep Biaffine Attention for Neural Dependency Parsing," *arXiv:1611.01734 [cs]*, Nov. 2016, arXiv: 1611.01734. [Online]. Available: http://arxiv.org/abs/1611.01734

[105] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[106] J. Hu, S. Ruder, A. Siddhant, G. Neubig, O. Firat, and M. Johnson, "XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalisation," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 4411–4421. [Online]. Available: http://proceedings.mlr.press/v119/hu20b.html

[107] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. [Online]. Available: https://aclanthology.org/N18-1101

[108] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[109] Y. Liu and M. Lapata, "Learning structured text representations," *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 63–75, 2018. [Online]. Available: https://www.aclweb.org/anthology/Q18-1005

[110] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A diversity-promoting objective function for neural conversation models," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 110–119. [Online]. Available: https://www.aclweb.org/anthology/N16-1014

[111] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. [Online]. Available: https://www.aclweb.org/anthology/D13-1170

[112] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 649–657.

[113] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[114] D. Yu, M. Wang, Y. Cao, I. Shafran, L. E. Shafey, and H. Soltau, "Unsupervised slot schema induction for task-oriented dialog," 2022.

[115] M. Eric, R. Goel, S. Paul, A. Sethi, S. Agarwal, S. Gao, A. Kumar, A. Goyal, P. Ku, and D. Hakkani-Tur, "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*. Marseille, France: European Language Resources Association, May 2020, pp. 422–428. [Online]. Available: https://aclanthology.org/2020.lrec-1.53

[116] X. Zang, A. Rastogi, S. Sunkara, R. Gupta, J. Zhang, and J. Chen, "MultiWOZ 2.2 : A dialogue dataset with additional annotation corrections and state tracking baselines," in *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*. Online: Association for Computational Linguistics, Jul. 2020, pp. 109–117. [Online]. Available: https://aclanthology.org/2020.nlp4convai-1.13

[117] Q. Min, L. Qin, Z. Teng, X. Liu, and Y. Zhang, "Dialogue state induction using neural latent variable models," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 3845–3852, main track. [Online]. Available: https://doi.org/10.24963/ijcai.2020/532

[118] D. Yu and Z. Yu, "MIDAS: A dialog act annotation scheme for open domain HumanMachine spoken conversations," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 1103–1120. [Online]. Available: https://aclanthology.org/2021.eacl-main.94

[119] Y.-N. Chen, W. Y. Wang, and A. I. Rudnicky, "Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013, pp. 120–125.

[120] V. Hudeček, O. Dušek, and Z. Yu, "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)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 2430–2442. [Online]. Available: https://aclanthology.org/2021.acl-long.189

[121] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gašić, "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*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 5016–5026. [Online]. Available: https://aclanthology.org/D18-1547

[122] A. Rastogi, X. Zang, S. Sunkara, R. Gupta, and P. Khaitan, "Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 8689–8696, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6394

[123] I. Klasinas, E. Iosif, K. Louka, and A. Potamianos, "SemEval-2014 task 2: Grammar induction for spoken dialogue systems," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, Aug. 2014, pp. 9–16. [Online]. Available: https://aclanthology.org/S14-2002

[124] G. Athanasopoulou, I. Klasinas, S. Georgiladakis, E. Iosif, and A. Potamianos, "Using lexical, syntactic and semantic features for non-terminal grammar rule induction in spoken dialogue systems," in *2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014, pp. 596–601.

[125] Y.-N. Chen, W. Y. Wang, and A. I. Rudnicky, "Leveraging frame semantics and distributional semantics for unsupervised semantic slot induction in spoken dialogue systems," in *2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014, pp. 584–589.

[126] ——, "Learning semantic hierarchy with distributed representations for unsupervised spoken language understanding," in *Proceedings of The 16th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2015)*, 2015, pp. 1869–1873.

[127] Y.-N. Chen, W. Y. Wang, and A. Rudnicky, "Jointly modeling inter-slot relations by random walk on knowledge graphs for unsupervised spoken language understanding," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, May–Jun. 2015, pp. 619–629. [Online]. Available: https://aclanthology.org/N15-1064

[128] C. F. Baker, C. J. Fillmore, and J. B. Lowe, "The Berkeley FrameNet project," in *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*, 1998. [Online]. Available: https://aclanthology.org/C98-1013

[129] D. Das, N. Schneider, D. Chen, and N. A. Smith, "Probabilistic frame-semantic parsing," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, Jun. 2010, pp. 948–956. [Online]. Available: https://aclanthology.org/N10-1138

[130] C. Shi, Q. Chen, L. Sha, S. Li, X. Sun, H. Wang, and L. Zhang, "Auto-dialabel: Labeling dialogue data with unsupervised learning," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 684–689. [Online]. Available: https://aclanthology.org/D18-1072

[131] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," in *Proceedings of the*

Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, 2017, pp. 1965–1972. [Online]. Available: https://doi.org/10.24963/ijcai.2017/273

[132] D. Yu, L. He, Y. Zhang, X. Du, P. Pasupat, and Q. Li, "Few-shot intent classification and slot filling with retrieved examples," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, Jun. 2021, pp. 734–749. [Online]. Available: https://aclanthology.org/2021.naacl-main.59

[133] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, and A. Jatowt, "Yake! collection-independent automatic keyword extractor," in *Advances in Information Retrieval*, G. Pasi, B. Piwowarski, L. Azzopardi, and A. Hanbury, Eds. Cham: Springer International Publishing, 2018, pp. 806–810.

[134] K. Bennani-Smires, C. Musat, A. Hossmann, M. Baeriswyl, and M. Jaggi, "Simple unsupervised keyphrase extraction using sentence embeddings," in *Proceedings of the 22nd Conference on Computational Natural Language Learning*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 221–229. [Online]. Available: https://aclanthology.org/K18-1022

[135] X. Wan and J. Xiao, "Single document keyphrase extraction using neighborhood knowledge," in *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, ser. AAAI'08. AAAI Press, 2008, p. 855–860.

[136] Z. Liu, P. Li, Y. Zheng, and M. Sun, "Clustering to find exemplar terms for keyphrase extraction," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore: Association for Computational Linguistics, Aug. 2009, pp. 257–266. [Online]. Available: https://aclanthology.org/D09-1027

[137] Y. Li, L. Liu, and K. Yao, "Neural sequence segmentation as determining the leftmost segments," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, Jun. 2021, pp. 1476–1486. [Online]. Available: https://aclanthology.org/2021.naacl-main.116

[138] D. Klein and C. D. Manning, "A generative constituent-context model for improved grammar induction," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, Jul. 2002, pp. 128–135. [Online]. Available: https://aclanthology.org/P02-1017

[139] D. Klein and C. Manning, "Corpus-based induction of syntactic structure: Models of dependency and constituency," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, Barcelona, Spain, Jul. 2004, pp. 478–485. [Online]. Available: https://aclanthology.org/P04-1061

[140] Y. Shen, Z. Lin, C. wei Huang, and A. Courville, "Neural language modeling by jointly learning syntax and lexicon," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=rkgOLb-0W

[141] A. Drozdov, P. Verga, Y.-P. Chen, M. Iyyer, and A. McCallum, "Unsupervised labeled parsing with deep inside-outside recursive autoencoders," 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1507–1512. [Online]. Available: https://aclanthology.org/D19-1161

[142] S. Zhang, L. Song, L. Jin, K. Xu, D. Yu, and J. Luo, "Video-aided unsupervised grammar induction," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, Jun. 2021, pp. 1513–1524. [Online]. Available: https://aclanthology.org/2021.naacl-main.119

[143] T. Kim, J. Choi, D. Edmiston, and S. goo Lee, "Are pre-trained language models aware of phrases? simple but strong baselines for grammar induction," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1xPR3NtPB

[144] J. Herzig and J. Berant, "Span-based semantic parsing for compositional generalization," 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)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 908–921. [Online]. Available: https://aclanthology.org/2021.acl-long.74

[145] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does BERT look at? an analysis of BERT's attention," in *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 276–286. [Online]. Available: https://aclanthology.org/W19-4828

[146] Y. Kim, C. Dyer, and A. Rush, "Compound probabilistic context-free grammars for grammar induction," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2369–2385. [Online]. Available: https://aclanthology.org/P19-1228

[147] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "SpanBERT: Improving pre-training by representing and predicting spans," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 64–77, 2020. [Online]. Available: https://aclanthology.org/2020.tacl-1.5

[148] C.-S. Wu, S. C. Hoi, R. Socher, and C. Xiong, "TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 917–929. [Online]. Available: https://aclanthology.org/2020.emnlp-main.66

[149] M. Henderson and I. Vulić, "ConVEx: Data-efficient and few-shot slot labeling," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, Jun. 2021, pp. 3375–3389. [Online]. Available: https://aclanthology.org/2021.naacl-main.264

[150] S. Davidson, D. Yu, and Z. Yu, "Dependency parsing for spoken dialog systems," 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1513–1519. [Online]. Available: https://aclanthology.org/D19-1162

[151] G. D. Forney, "The viterbi algorithm," *Proc. of the IEEE*, vol. 61, pp. 268 – 278, March 1973.

[152] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017. [Online]. Available: https://doi.org/10.21105/joss.00205

[153] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0377042787901257

[154] K. Yamada, R. Sasano, and K. Takeda, "Semantic frame induction using masked word embeddings and two-step clustering," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 811–816. [Online]. Available: https://aclanthology.org/2021.acl-short.102

[155] J. Michael, J. A. Botha, and I. Tenney, "Asking without telling: Exploring latent ontologies in contextual representations," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 6792–6812. [Online]. Available: https://aclanthology.org/2020.emnlp-main.552

[156] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The Stanford CoreNLP natural language processing toolkit," in *Proceedings of*

52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 55–60. [Online]. Available: https://aclanthology.org/P14-5010

[157] W. Lei, X. Jin, M.-Y. Kan, Z. Ren, X. He, and D. Yin, "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)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1437–1447. [Online]. Available: https://aclanthology.org/P18-1133

[158] Y. Zhang, Z. Ou, and Z. Yu, "Task-oriented dialog systems that consider multiple appropriate responses under the same context," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 9604–9611, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6507

[159] E. Hosseini-Asl, B. McCann, C.-S. Wu, S. Yavuz, and R. Socher, "A simple language model for task-oriented dialogue," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 20 179–20 191. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/e946209592563be0f01c844ab2170f0c-Paper.pdf

[160] W. Chen, J. Chen, P. Qin, X. Yan, and W. Y. Wang, "Semantically conditioned dialog response generation via hierarchical disentangled self-attention," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3696–3709. [Online]. Available: https://aclanthology.org/P19-1360

[161] X. Du, L. He, Q. Li, D. Yu, P. Pasupat, and Y. Zhang, "QA-driven zero-shot slot filling with weak supervision pretraining," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 654–664. [Online]. Available: https://aclanthology.org/2021.acl-short.83

[162] H. Shi, J. Mao, K. Gimpel, and K. Livescu, "Visually grounded neural syntax acquisition," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1842–1861. [Online]. Available: https://aclanthology.org/P19-1180

[163] R. Lowe, N. Pow, I. Serban, and J. Pineau, "The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems," in *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Prague, Czech Republic: Association for Computational Linguistics, Sep. 2015, pp. 285–294. [Online]. Available: https://aclanthology.org/W15-4640

[164] S. Min, D. Chen, H. Hajishirzi, and L. Zettlemoyer, "A discrete hard EM approach for weakly supervised question answering," 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2851–2864. [Online]. Available: https://aclanthology.org/D19-1284

[165] D. Yu and K. Sagae, "Automatically exposing problems with neural dialog models," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 456–470. [Online]. Available: https://aclanthology.org/2021.emnlp-main.37

[166] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4077–4087. [Online]. Available: http://papers.nips.cc/paper/6996-prototypical-networks-for-few-shot-learning.pdf

[167] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P. Manzagol, and H. Larochelle, "Meta-dataset: A dataset of datasets for learning to learn from few examples," *CoRR*, vol. abs/1903.03096, 2019. [Online]. Available: http://arxiv.org/abs/1903.03096

[168] C.-S. Wu, A. Madotto, E. Hosseini-Asl, C. Xiong, R. Socher, and P. Fung, "Transferable multi-domain state generator for task-oriented dialogue systems," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 808–819. [Online]. Available: https://aclanthology.org/P19-1078

[169] K. Q. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds., vol. 18. MIT Press, 2006, pp. 1473–1480. [Online]. Available: https://proceedings.neurips.cc/paper/2005/file/a7f592cef8b130a6967a90617db5681b-Paper.pdf

[170] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 3630–3638. [Online]. Available: http://papers.nips.cc/paper/6385-matching-networks-for-one-shot-learning.pdf

[171] S. Wiseman and K. Stratos, "Label-agnostic sequence labeling by copying nearest neighbors," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5363–5369. [Online]. Available: https://aclanthology.org/P19-1533

[172] Y. Hou, W. Che, Y. Lai, Z. Zhou, Y. Liu, H. Liu, and T. Liu, "Few-shot slot tagging with collapsed dependency transfer and label-enhanced task-adaptive projection network," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 1381–1393. [Online]. Available: https://aclanthology.org/2020.acl-main.128

[173] S. Toshniwal, H. Shi, B. Shi, L. Gao, K. Livescu, and K. Gimpel, "A cross-task analysis of text span representations," in *Proceedings of the 5th Workshop on Representation Learning for NLP*. Online: Association for Computational Linguistics, Jul. 2020, pp. 166–176. [Online]. Available: https://aclanthology.org/2020.repl4nlp-1.20

[174] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[175] K. Roth, T. Milbich, S. Sinha, P. Gupta, B. Ommer, and J. P. Cohen, "Revisiting training strategies and generalization performance in deep metric learning," 2020.

[176] Y. Yang, G. Hernandez Abrego, S. Yuan, M. Guo, Q. Shen, D. Cer, Y.-h. Sung, B. Strope, and R. Kurzweil, "Improving multilingual sentence embedding using bi-directional dual encoder with additive margin softmax," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 5370–5378. [Online]. Available: https://doi.org/10.24963/ijcai.2019/746

[177] S. Larson, A. Mahendran, J. J. Peper, C. Clarke, A. Lee, P. Hill, J. K. Kummerfeld, K. Leach, M. A. Laurenzano, L. Tang, and J. Mars, "An evaluation dataset for intent classification and out-of-scope prediction," 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1311–1316. [Online]. Available: https://aclanthology.org/D19-1131

[178] A. Coucke, A. Saade, A. Ball, T. Bluche, A. Caulier, D. Leroy, C. Doumouro, T. Gisselbrecht, F. Caltagirone, T. Lavril, M. Primet, and J. Dureau, "Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces," 2018.

[179] J. Krone, Y. Zhang, and M. Diab, "Learning to classify intents and slot labels given a handful of examples," in *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*. Online: Association for Computational Linguistics, Jul. 2020, pp. 96–108. [Online]. Available: https://aclanthology.org/2020.nlp4convai-1.12

[180] S. W. Yoon, J. Seo, and J. Moon, "TapNet: Neural network augmented with task-adaptive projection for few-shot learning," in *Proceedings of Machine Learning Research*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 7115–7123. [Online]. Available: http://proceedings.mlr.press/v97/yoon19a.html

[181] S. Humeau, K. Shuster, M. Lachaux, and J. Weston, "Real-time inference in multi-sentence tasks with deep pretrained transformers," *CoRR*, vol. abs/1905.01969, 2019. [Online]. Available: http://arxiv.org/abs/1905.01969

[182] G. R. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning workshop*, 2015.

[183] D. Gillick, S. Kulkarni, L. Lansing, A. Presta, J. Baldridge, E. Ie, and D. Garcia-Olano, "Learning dense representations for entity retrieval," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 528–537. [Online]. Available: https://aclanthology.org/K19-1049

[184] R. Geng, B. Li, Y. Li, X. Zhu, P. Jian, and J. Sun, "Induction networks for few-shot text classification," 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3904–3913. [Online]. Available: https://aclanthology.org/D19-1403

[185] M. Yu, X. Guo, J. Yi, S. Chang, S. Potdar, Y. Cheng, G. Tesauro, H. Wang, and B. Zhou, "Diverse few-shot text classification with multiple metrics," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1206–1215. [Online]. Available: https://aclanthology.org/N18-1109

[186] S. Sun, Q. Sun, K. Zhou, and T. Lv, "Hierarchical attention prototypical networks for few-shot text classification," 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 476–485. [Online]. Available: https://aclanthology.org/D19-1045

[187] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis, "Generalization through memorization: Nearest neighbor language models," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HklBjCEKvH

[188] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M.-W. Chang, "Realm: Retrieval-augmented language model pre-training," 2020.

[189] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2020.

[190] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. ICML'17, vol. 70. JMLR.org, 2017, pp. 1126–1135. [Online]. Available: https://dl.acm.org/doi/10.5555/3305381.3305498

[191] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=B1DmUzWAW

[192] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," *CoRR*, vol. abs/1508.01991, 2015. [Online]. Available: http://arxiv.org/abs/1508.01991

[193] M. Ziyadi, Y. Sun, A. Goswami, J. Huang, and W. Chen, "Example-based named entity recognition," 2020.

[194] Y. Zhang, S. Sun, M. Galley, Y. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan, "Dialogpt: Large-scale generative pre-training for conversational response generation," *CoRR*, vol. abs/1911.00536, 2019. [Online]. Available: http://arxiv.org/abs/1911.00536

[195] D. Adiwardana, M. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, and Q. V. Le, "Towards a human-like open-domain chatbot," *CoRR*, vol. abs/2001.09977, 2020. [Online]. Available: https://arxiv.org/abs/2001.09977

[196] S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, E. M. Smith, Y.-L. Boureau, and J. Weston, "Recipes for building an open-domain chatbot," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 300–325. [Online]. Available: https://aclanthology.org/2021.eacl-main.24

[197] J. Li, M. Galley, C. Brockett, G. Spithourakis, J. Gao, and B. Dolan, "A persona-based neural conversation model," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 994–1003. [Online]. Available: https://aclanthology.org/P16-1094

[198] S. Welleck, I. Kulikov, S. Roller, E. Dinan, K. Cho, and J. Weston, "Neural text generation with unlikelihood training," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SJeYe0NtvH

[199] M. Li, S. Roller, I. Kulikov, S. Welleck, Y.-L. Boureau, K. Cho, and J. Weston, "Don't say that! making inconsistent dialogue unlikely with unlikelihood training," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 4715–4728. [Online]. Available: https://aclanthology.org/2020.acl-main.428

[200] C. Xu, W. Zhou, T. Ge, K. Xu, J. McAuley, and F. Wei, "Beyond preserved accuracy: Evaluating loyalty and robustness of BERT compression," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 10 653–10 659. [Online]. Available: https://aclanthology.org/2021.emnlp-main.832

[201] D. Yu, M. Cohn, Y. M. Yang, C. Y. Chen, W. Wen, J. Zhang, M. Zhou, K. Jesse, A. Chau, A. Bhowmick, S. Iyer, G. Sreenivasulu, S. Davidson, A. Bhandare, and Z. Yu, "Gunrock: A social bot for complex and engaging long conversations," 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): System Demonstrations*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 79–84. [Online]. Available: https://aclanthology.org/D19-3014

[202] A. Paranjape, A. See, K. Kenealy, H. Li, A. Hardy, P. Qi, K. R. Sadagopan, N. M. Phu, D. Soylu, and C. D. Manning, "Neural generation meets real people: Towards emotionally engaging mixed-initiative conversations," *CoRR*, vol. abs/2008.12348, 2020. [Online]. Available: https://arxiv.org/abs/2008.12348

[203] M. J. Wolf, K. Miller, and F. S. Grodzinsky, "Why we should have seen that coming: Comments on microsoft's tay "experiment," and wider implications," *SIGCAS Comput. Soc.*, vol. 47, no. 3, p. 54–64, Sep. 2017. [Online]. Available: https://doi.org/10.1145/3144592.3144598

[204] T. Simonite, "It began as an ai-fueled dungeon game. it got much darker," May 2021. [Online]. Available: https://www.wired.com/story/ai-fueled-dungeon-game-got-much-darker/

[205] J. Xu, D. Ju, M. Li, Y. Boureau, J. Weston, and E. Dinan, "Recipes for safety in open-domain chatbots," *CoRR*, vol. abs/2010.07079, 2020. [Online]. Available: https://arxiv.org/abs/2010.07079

[206] Y. Nie, M. Williamson, M. Bansal, D. Kiela, and J. Weston, "I like fish, especially dolphins: Addressing contradictions in dialogue modeling," 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). Online: Association for Computational Linguistics, Aug. 2021, pp. 1699–1713. [Online]. Available: https://aclanthology.org/2021.acl-long.134

[207] A. Radford, R. Józefowicz, and I. Sutskever, "Learning to generate reviews and discovering sentiment," *CoRR*, vol. abs/1704.01444, 2017. [Online]. Available: http://arxiv.org/abs/1704.01444

[208] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," 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)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 4582–4597. [Online]. Available: https://aclanthology.org/2021.acl-long.353

[209] D. Yu, K. Sagae, and Z. Yu, "Attribute alignment: Controlling text generation from pre-trained language models," *CoRR*, vol. abs/2103.11070, 2021. [Online]. Available: https://arxiv.org/abs/2103.11070

[210] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, and R. Liu, "Plug and play language models: A simple approach to controlled text generation," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1edEyBKDS

[211] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *International Conference on Learning Representations*, 2017. [Online]. Available: http://arxiv.org/abs/1611.00712

[212] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://arxiv.org/abs/1611.01144

[213] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[214] J. Baumgartner, S. Zannettou, B. Keegan, M. Squire, and J. Blackburn, "The pushshift reddit dataset," *CoRR*, vol. abs/2001.08435, 2020. [Online]. Available: https://arxiv.org/abs/2001.08435

[215] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, "Personalizing dialogue agents: I have a dog, do you have pets too?" in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 2204–2213. [Online]. Available: https://aclanthology.org/P18-1205

[216] E. M. Smith, M. Williamson, K. Shuster, J. Weston, and Y.-L. Boureau, "Can you put it all together: Evaluating conversational agents' ability to blend skills," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 2021–2030. [Online]. Available: https://aclanthology.org/2020.acl-main.183

[217] E. Dinan, S. Humeau, B. Chintagunta, and J. Weston, "Build it break it fix it for dialogue safety: Robustness from adversarial human attack," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 4537–4546. [Online]. Available: https://aclanthology.org/D19-1461

[218] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[219] Y. Nie, A. Williams, E. Dinan, M. Bansal, J. Weston, and D. Kiela, "Adversarial NLI: A new benchmark for natural language understanding," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 4885–4901. [Online]. Available: https://aclanthology.org/2020.acl-main.441

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

[221] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 8342–8360. [Online]. Available: https://aclanthology.org/2020.acl-main.740

[222] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh, "Universal adversarial triggers for attacking and analyzing NLP," 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 2153–2162. [Online]. Available: https://aclanthology.org/D19-1221

[223] J. Ficler and Y. Goldberg, "Controlling linguistic style aspects in neural language generation," in *Proceedings of the Workshop on Stylistic Variation*. Copenhagen,

Denmark: Association for Computational Linguistics, Sep. 2017, pp. 94–104. [Online]. Available: https://aclanthology.org/W17-4912

[224] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, "CTRL: A conditional transformer language model for controllable generation," *CoRR*, vol. abs/1909.05858, 2019. [Online]. Available: http://arxiv.org/abs/1909.05858

[225] B. Peng, C. Zhu, C. Li, X. Li, J. Li, M. Zeng, and J. Gao, "Few-shot natural language generation for task-oriented dialog," in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 172–182. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.17

[226] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *Proceedings of Machine Learning Research*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 2790–2799. [Online]. Available: http://proceedings.mlr.press/v97/houlsby19a.html

[227] J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao, "Deep reinforcement learning for dialogue generation," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1192–1202. [Online]. Available: https://aclanthology.org/D16-1127

[228] I. V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N. R. Ke, S. Mudumba, A. de Brébisson, J. Sotelo, D. Suhubdy, V. Michalski, A. Nguyen, J. Pineau, and Y. Bengio, "A deep reinforcement learning chatbot," *CoRR*, vol. abs/1709.02349, 2017. [Online]. Available: http://arxiv.org/abs/1709.02349

[229] Y. Zhang, J. Baldridge, and L. He, "PAWS: Paraphrase adversaries from word scrambling," 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 1298–1308. [Online]. Available: https://aclanthology.org/N19-1131

[230] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is bert really robust? a strong baseline for natural language attack on text classification and entailment," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 8018–8025, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6311

[231] D. Li, Y. Zhang, H. Peng, L. Chen, C. Brockett, M.-T. Sun, and B. Dolan, "Contextualized perturbation for textual adversarial attack," in *Proceedings of*

the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Online: Association for Computational Linguistics, Jun. 2021, pp. 5053–5069. [Online]. Available: https://aclanthology.org/2021.naacl-main.400

[232] L. Song, X. Yu, H.-T. Peng, and K. Narasimhan, "Universal adversarial attacks with natural triggers for text classification," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, Jun. 2021, pp. 3724–3733. [Online]. Available: https://aclanthology.org/2021.naacl-main.291

[233] E. Sheng, K.-W. Chang, P. Natarajan, and N. Peng, "Towards Controllable Biases in Language Generation," in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 3239–3254. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.291

[234] S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith, "RealToxicityPrompts: Evaluating neural toxic degeneration in language models," in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 3356–3369. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.301

[235] T. He and J. R. Glass, "Detecting egregious responses in neural sequence-to-sequence models," *CoRR*, vol. abs/1809.04113, 2018. [Online]. Available: http://arxiv.org/abs/1809.04113

[236] H. Liu, T. Derr, Z. Liu, and J. Tang, "Say what I want: Towards the dark side of neural dialogue models," *CoRR*, vol. abs/1909.06044, 2019. [Online]. Available: http://arxiv.org/abs/1909.06044

[237] H. Liu, Z. Wang, T. Derr, and J. Tang, "Chat as expected: Learning to manipulate black-box neural dialogue models," *CoRR*, vol. abs/2005.13170, 2020. [Online]. Available: https://arxiv.org/abs/2005.13170

[238] T. He and J. Glass, "Negative training for neural dialogue response generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 2044–2058. [Online]. Available: https://aclanthology.org/2020.acl-main.185

[239] M. Zampieri, P. Nakov, S. Rosenthal, P. Atanasova, G. Karadzhov, H. Mubarak, L. Derczynski, Z. Pitenis, and Ç. Çöltekin, "SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020)," in *Proceedings of the Fourteenth Workshop on Semantic Evaluation*. Barcelona (online): International Committee for Computational Linguistics, Dec. 2020, pp. 1425–1447. [Online]. Available: https://aclanthology.org/2020.semeval-1.188

[240] E. Dinan, A. Fan, L. Wu, J. Weston, D. Kiela, and A. Williams, "Multi-dimensional gender bias classification," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 314–331. [Online]. Available: https://aclanthology.org/2020.emnlp-main.23

[241] S. Sukhbaatar, a. szlam, J. Weston, and R. Fergus, "End-to-end memory networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/8fb21ee7a2207526da55a679f0332de2-Paper.pdf

[242] N. Dziri, E. Kamalloo, K. Mathewson, and O. Zaiane, "Evaluating coherence in dialogue systems using entailment," in *Proceedings of the 2019 Workshop on Widening NLP*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 146–148. [Online]. Available: https://aclanthology.org/W19-3646

[243] S. Welleck, J. Weston, A. Szlam, and K. Cho, "Dialogue natural language inference," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 3731–3741. [Online]. Available: https://aclanthology.org/P19-1363

[244] C.-Y. Chen, D. Yu, W. Wen, Y. M. Yang, J. Zhang, M. Zhou, K. Jesse, A. Chau, A. Bhowmick, S. Iyer, G. Sreenivasulu, R. Cheng, A. Bhandare, and Z. Yu, "Gunrock: Building a human-like social bot by leveraging large scale real user data," in *2nd Proceedings of Alexa Prize*, 2018.

[245] K. Liang, A. Chau, Y. Li, X. Lu, D. Yu, M. Zhou, I. Jain, S. Davidson, J. Arnold, M. Nguyen, and Z. Yu, "Gunrock 2.0: A user adaptive social conversational system," *CoRR*, vol. abs/2011.08906, 2020. [Online]. Available: https://arxiv.org/abs/2011.08906

[246] X. Zhang, C. Li, D. Yu, S. Davidson, and Z. Yu, "Filling conversation ellipsis for better social dialog understanding," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 9587–9595, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6505

[247] D. Yu and K. Sagae, "UC Davis at SemEval-2019 task 1: DAG semantic parsing with attention-based decoder," in *Proceedings of the 13th International Workshop on Semantic Evaluation*. Minneapolis, Minnesota, USA: Association for Computational Linguistics, Jun. 2019, pp. 119–124. [Online]. Available: https://aclanthology.org/S19-2017

[248] Y. Du, Q. Fang, and D. Nguyen, "Assessing the reliability of word embedding gender bias measures," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic:

Association for Computational Linguistics, Nov. 2021, pp. 10 012–10 034. [Online]. Available: https://aclanthology.org/2021.emnlp-main.785

[249] J. Zhao, R. Gupta, Y. Cao, D. Yu, M. Wang, H. Lee, A. Rastogi, I. Shafran, and Y. Wu, "Description-driven task-oriented dialog modeling," *CoRR*, vol. abs/2201.08904, 2022. [Online]. Available: https://arxiv.org/abs/2201.08904

[250] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre, "Improving language models by retrieving from trillions of tokens," *CoRR*, vol. abs/2112.04426, 2021. [Online]. Available: https://arxiv.org/abs/2112.04426

[251] E. Perez, S. Huang, H. F. Song, T. Cai, R. Ring, J. Aslanides, A. Glaese, N. McAleese, and G. Irving, "Red teaming language models with language models," *CoRR*, vol. abs/2202.03286, 2022. [Online]. Available: https://arxiv.org/abs/2202.03286