Symbolic Dialogue for General Domain State Tracking

by

Nguyen Son Tung, Nguyen Manh Tuong, Le Xuan Tung

FPT UNIVERSITY

Symbolic Dialogue for General Domain State Tracking

by
Nguyen Son Tung, Nguyen Manh Tuong, Le Xuan Tung

Supervisor: Nguyen Quoc Trung

A final year capstone project submitted in partial fulfillment of the requirement for the Degree of Bachelor of Artificial Intelligent in Computer Science

ACKNOWLEDGMENTS

In this section, we would like to acknowledge the following individuals and organizations for their valuable contributions and support:

- Our supervisor, Nguyen Quoc Trung, for his guidance, expertise, and encouragement throughout this project.
- Gradient Technology .inc for the great support from idealization to training process, research review and submission.
- We especially thank Mr. Tien - Gradient Technology .inc - for his dedication to the project. His valuable opinion is the main contribution to the success of this project.
- Our friends and family for their love and support throughout our academic journey.

We are grateful to all those who have helped us in ways both big and small, and without whom this project would not have been possible. Thank you all.

ABSTRACT

Task-oriented dialogue (TOD) is a system that helps users achieve their goals. While the task is reviewed and improved regularly, a formal system for industrial standards has not yet been established. Dialogue state tracking is a sub-task that involves predicting current dialogue slot values given the conversation and in some cases, the slots that are being required or informed. Based on a well-documented schema with instructions for possible slots and intents along with their descriptions, schema-guided TOD exploits a concrete set of guidelines to add extra context and perform general zero-shot ability on state tracking. Despite having contextual schema descriptions, language models hardly keep up with a full TOD dialogue flow. The TOD system as a whole lacks the mechanics to detect out-of-scope events, decide when to query the database, and is hardly extensible for further processing. To address these issues, we propose a full TOD system designed to overcome the listed weaknesses. Additionally, we experiment with dialogue state tracking, the system's first stage, and measure out-of-scope detection effectiveness via user-undefined actions.

Index Terms—task-oriented dialogue, schema-guided, symbolic reasoning, open-domain dialogue, dialogue system, schema actions

TABLE OF CONTENT

LIST OF FIGURES

I. INTRODUCTION

The ability to work as a guide is essential for AI to support human daily chores. The TOD system focuses on helping the user achieve their goals by having an instructive conversation. Unfortunately, in reality, not every utterance is direct and precise. In general, there are cases where the user might provide the wrong context or go out of scope of the current context. In the TOD realm, out-of-scope represents both domain switching and inconsequential conversation. Take customer support; for example, workers must learn to understand company products and the context of a situation. Despite the preparation, workers are reactive only to situations within their knowledge. If the situation is out-of-scope, the worker has to review the guidelines or pass the session to a specialized department. The goal is to leave no case unsupported, thus elevating the customer support experience. With human-level consciousness, the out-of-scope situation is resolvable by utilizing external sources. Handling cases out-of-scope is a necessity for a TOD system to reach human-level consciousness and cover all cases.

Mimicking human behavior, the schema-guided method uses domain schema to understand the conversation. Domain schema provides information such as descriptions, possible values, slots, and intents. Using the schema as a guideline helps generalize and yield remarkable performance on state tracking. Yet, only maintaining the dialogue state is extensible, as the state cuts out the crucial context of the dialogue, limiting further processes. Additionally, the schema-based method is incapable of extending or applying unstructured data. To address these issues, the schema-based system must output a sufficient amount of information and be capable of handling out-of-scope situations.

In recent years, large language models (LLM) have exponentially grown in number. Consequently, LLM-based TOD systems introduce end-to-end methods that dominate academic benchmarks. However, the end-to-end method often suffers from dramatic domain changes and lengthy input contexts. During inference, lengthy input dialogue may overwhelm the LLM, leading to poor performance. In Schema-Guided Dialogue (SGD), a conversation could reach 23 utterances, whereas in an industry context, the number can be much higher. State tracking consists of sub-tasks, namely, slot filling, user request slots, and other optional tasks. While in slot filling, the lengthy conversation as input is inevitable because of the state value extraction. Other tasks of state tracking can exploit the LLM symbolic method to eliminate conversation dependency. The symbolic method is a recently developed technique for more fine-grained control over LLM output.

Previous works proved the symbolic method can improve performance on language reasoning-related tasks. Traditional deep learning lets the model discover hidden patterns through implicit exploration. While the symbolic method is more direct by explicitly stating the pattern via the symbolizing process.

We propose a hybrid TOD framework that combines schema-guided, retrieval-augmented, and symbolic. In this approach, the schema is used to guide the language model and prevent it from going off-track. While the symbolic method acts as the mechanic to control the system flow. Additionally, the retrieval model is used to retrieve relevant information from external sources in case of out-of-scope situations. Notice in the method that we use a single encoder-decoder model for every state tracking-related task to give the model a wider view of the problem. State tracking is the most important module of the system as it's precision directly affects downstream tasks. We implement and report the results of state tracking with out-of-scope mechanics. We additionally provide more context to the model with auxiliary tags and symbolize slot values.

II. RELATED WORKS

1. Background of Task-oriented dialogue

State tracking traditionally uses encoder architecture as a classifier for every individual sub-task. Despite comparative performance, encoder-based state tracking modules are either inefficient or complex [1] [2]. Previously, the encoder for TOD was categorized as single pass and multipass. FastSGT is a single pass encoder, while being efficient, the performance is not as impressive as other multipass models. In encoder architecture, although a pre-trained process is made, researchers use the model for transfer learning by appending untrained parameters to the model. The complete model is then fine-tuned for State tracking. Thus, the ability learned during pre-trained is not fully utilized by the supplemental parameters. Subsequently, researchers develop in-context learning [3] which utilizes even further grounded knowledge. In-context learning supports the model by having more explicit context at different levels. Note that in-context learning is frequently used for text-based output modality models. Since then, two directions have come up, namely, end-to-end and modular, both utilizing the transformer deep learning model. End-to-end TOD systems work by iteratively concatenating the answer from the previous turn to the context. End-to-end systems suffer from lengthy input, and accumulative error while facing strong coupling. In software design, modular approaches are preferred due to the ability for independent improvement. Modular approaches, alternatively, use transformers only to predict a single stage of TOD-state tracking. Language models perform excellently on state tracking, even in zero-shot settings. Unlike previous token-based slot-capturing encoder models, the language model was pre-trained to perform token prediction, similar to state tracking.

End-to-end method

SimpleTOD [4] is a single, unified language model capable of all TOD sub-tasks (dialogue state tracking, action decision, and response generation) using GPT- 2. With SimpleTOD, inference is a simple, iterative process and uses every possible context to support prediction. SimpleTOD is inefficient in terms of computation and is trained with MultiWOZ [5]. MultiWOZ is a multi-domain dataset with similar domain clusters which is inefficient for zero shot business use cases. SPACE-3 [6] follows an end-to-end paradigm, but the model includes multiple decoders for each TOD task. SPACE-3 also applies a multi-task paradigm with a clear pre-train process. Unlike SPACE-3, SimpleTOD does not have a step between pre-training and fine-tuning. Alternatively, there is a vast majority of branches to further improve the performance of end-to-end models. These include enhanced schema robustness [7] [8], TOD adapter architecture [9], LLM with schema support [10],…

TOD with open-domain

From the human perspective, the conversation is not only about goal achievement but also about gaining social support. Thus, preventing tasks from becoming overwhelming in TOD, recently, gained a significant amount of priority. UniDS [11] deals with chit-chat by sequentially training a model with chit-chat situations, then, TOD dataset. Even with open-domain, data is rapidly shifting, and obsolete data used during training. OPERA [12] introduces knowledge grounding as a mechanic to accurately reply to open domain utterances, which makes it a robust solution for both question-answering and TOD tasks. Both successful models and improvement in dataset [13] suggest open-domains should employ retrieval models as knowledge validators. Nonetheless, the works mentioned are end-to-end and did not have a clear mechanism to change between TOD and open-domain.

Language model for state tracking

Language models treat state-tracking problems simply as multi-QA problems. Given the information about slots and the conversation, the language model can name all slots available in the conversation along with their corresponding values. Moreover, only slot filling is not sufficient for further TOD tasks, state tracking with language model must also do request-inform actions tracking and intent tracking. In D3ST [14], the given information is the description of the slots and intents. SDT [15] approach is slightly different from D3ST, the model is given a demonstration process of state tracking and asked to revise with the real conversation. While SDT outperformed D3ST, the module is hard to extend because of the compulsory n-shot learning attribute. Overall, these models are innovative, yet, facing an extreme challenge: action unification. Across TOD datasets, the actions for the user and system on each dataset are different. The follow-up process of state tracking is a dialogue policy. As a fact, we need a robust set of actions. Still, the actions to get a robust set of policies. There are prior works addressing the issue by several methods i.e. unified action dataset [7], and extract action latent space [16] [17]. The unified action dataset is interpretable but cost-flexible, the contrast happened to extract action latent space.

Symbolic methods

Humans in some cases can understand problems better when they involve symbols [18]. [19] experimented with symbolic methods on LLM for mathematics. Agree on symbols introduces new abstract meaning to the symbols, thus shortening the explaining process. Several studies suggest symbolic reasoning [20] [21] [22] can also improve model performance on reasoning tasks. In fact, SDT and D3ST implicitly use symbolic methods to reach state-of-art performance. In technical terms, the input and output of the language model involve well-trained tokens, the probability of one token is not only peer-dependent but highly correlated to the previous pre-train. By symbolizing tokens, the model can learn to associate tokens with more general concepts. This, hypothetically, makes the model more likely to generalize better to new data, even if that data is from a different domain. By using the symbolic method, AnyTOD [23] combines the language model and flexible action set by explicitly adding possible actions to the input prompt. Although AnyTOD performs outstanding on multiple academic datasets, the proposed system did not mention mechanics due to abnormal situations in the conversation.

2. *Background of modern AI*

RNNs

Recurrent Neural Networks (RNNs) are a type of neural network architecture that is particularly well-suited for processing sequential data. Unlike feedforward neural networks, which process inputs in isolation, RNNs have a built-in feedback mechanism that allows them to maintain an internal state or memory of past inputs. This memory enables RNNs to capture dependencies and patterns in sequential data, making them powerful tools for tasks such as natural language processing, speech recognition, and time series analysis.

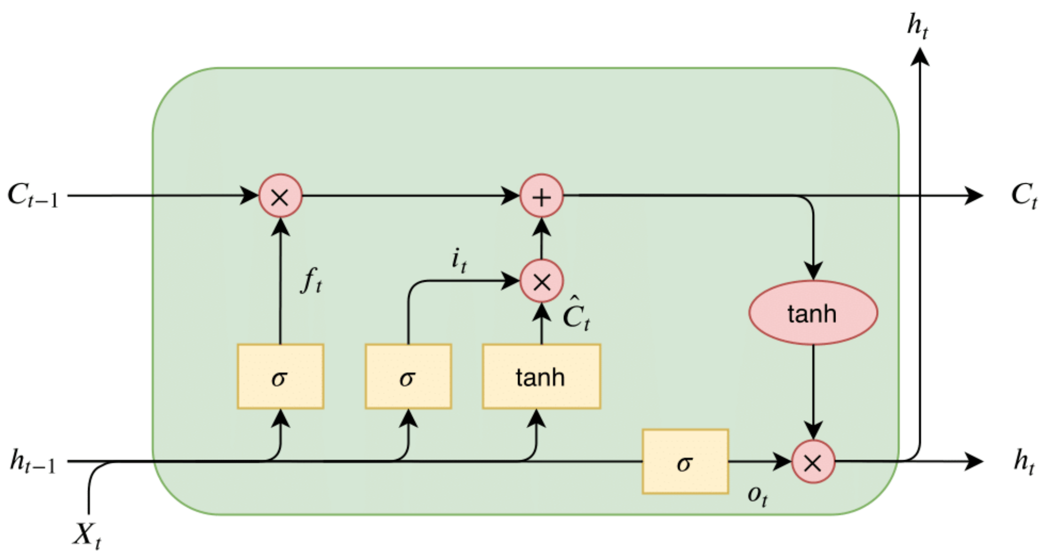

Figure 1a. Illustration of RNN achitecture
(https://thorirmar.com/post/insight_into_lstm)

The key idea behind RNNs is the concept of recurrence. At each time step, an RNN takes an input and combines it with the internal state from the previous time step to produce an output and update its internal state. This feedback loop allows the RNN to incorporate information from previous inputs and make predictions based on the context it has learned so far. Mathematically, an RNN can be represented as a series of interconnected neurons, where each neuron takes an input, produces an output, and passes its output to the next neuron in the sequence. The output of each neuron is determined by a combination of the current input and the output of the previous neuron. This recursive relationship allows the RNN to propagate information through time. One of the key advantages of RNNs is their ability to handle variable-length sequences. Unlike traditional feedforward networks, which require fixed-size inputs, RNNs can process inputs of different lengths by unrolling the sequence over time. This flexibility makes RNNs well-suited for tasks such as sentiment analysis, machine translation, and speech recognition, where the length of the input can vary. However, traditional RNNs suffer from the ”vanishing gradient” problem, which limits their ability to capture long-term dependencies in sequential data. When training an RNN using gradient-based methods, the gradients tend to either explode or diminish exponentially as they propagate through time, leading to difficulties in learning long range dependencies. To address this issue, several variants of RNNs have been developed, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). LSTMs and GRUs are designed to mitigate the vanishing gradient problem by introducing gating mechanisms that control the flow of information within the network. These gates determine how much information from the current input and the previous state should be passed along to the next time step. By selectively updating and forgetting information, LSTMs, and GRUs can effectively learn and retain long-term dependencies in the data. RNNs can be trained using backpropagation through time (BPTT), an extension of the standard backpropagation algorithm. BPTT unfolds the recurrent connections in time and computes the gradients at each time step, allowing the network to learn from past inputs and update its internal parameters accordingly. The training process involves minimizing a loss function that measures the discrepancy between the predicted outputs and the true outputs. In recent years, RNNs have been widely applied in various domains. In natural language processing, RNNs have been used for tasks such as language modeling, text generation, and sentiment analysis. In speech recognition, RNNs have demonstrated their effectiveness in modeling sequential audio data. RNNs have also shown promise in time series analysis, where they can capture temporal dependencies and make predictions based on historical patterns.

In conclusion, Recurrent Neural Networks (RNNs) are a class of neural network architectures that excel at processing sequential data. Their ability to maintain an internal state and capture dependencies across time makes them well-suited for a wide range of applications. Despite the challenges associated with training and the vanishing gradient problem, variants such as LSTMs and GRUs have significantly improved the performance of RNNs. As research in deep learning continues to advance, RNNs are likely to remain a fundamental tool for modeling and understanding sequential data.

CNNs

Convolutional Neural Networks (CNNs) are a powerful class of deep learning models that have revolutionized the field of computer vision. CNNs are specifically designed to process and analyze visual data, such as images and videos. They have proven to be highly effective in tasks such as image classification, object detection, and image segmentation.

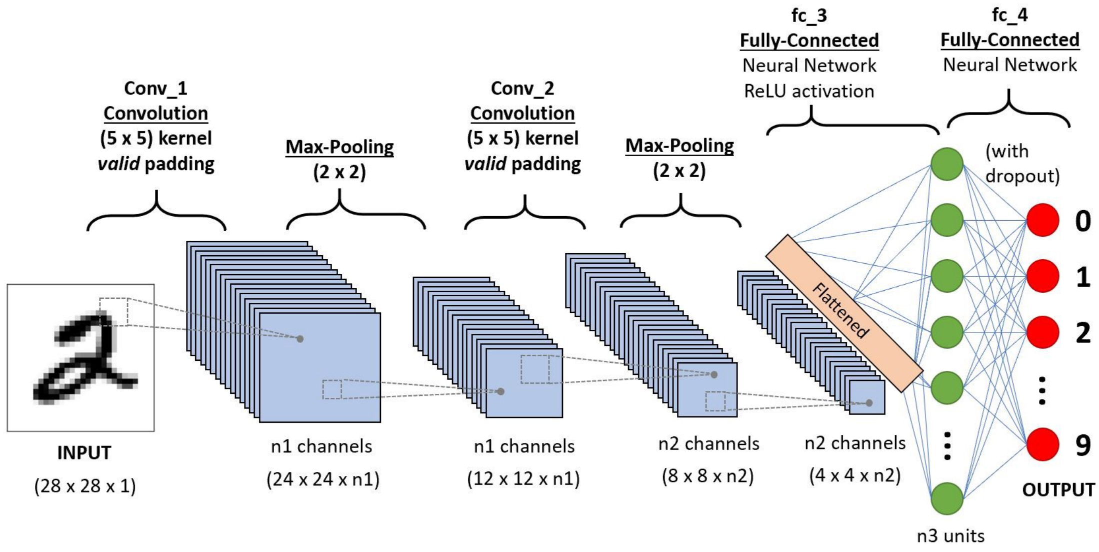

Figure 1b. Illustration of CNN achitecture
(https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/)

The key idea behind CNNs is the concept of convolution. Convolution is a mathematical operation that involves applying a filter or kernel to an input image to extract local features. These filters slide over the input image, computing a dot product between the filter weights and the corresponding pixel values at each location. The resulting output, known as a feature map, highlights the presence of specific features in the input image. CNNs consist of multiple layers, each with a specific purpose. The first layer is the input layer, which receives the raw pixel values of the image. Typically, the input image is preprocessed by resizing it to a fixed size and normalizing the pixel values. This ensures that the input is consistent across different images. The subsequent layers of a CNN are composed of convolutional layers, pooling layers, and fully connected layers. Convolutional layers are responsible for learning and extracting local features from the input image. They consist of multiple filters, each detecting a different feature or pattern. During training, the CNN automatically learns the optimal filter weights that maximize the detection of relevant features. Pooling layers are inserted between convolutional layers to reduce the spatial dimensions of the feature maps. Pooling operations, such as max pooling, average pooling, or sum pooling, aggregate the information within a local neighborhood, reducing the computational complexity and extracting the most important features. After several convolutional and pooling layers, the last part of the CNN is typically composed of fully connected layers. These layers are similar to those found in traditional neural networks and are responsible for making the final predictions based on the extracted features. Fully connected layers combine the features learned from the previous layers and map them to the output classes or labels. One of the major advantages of CNNs is their ability to capture spatial hierarchies of features. The initial layers of the network learn simple and low-level features, such as edges and textures, while deeper layers learn more complex and high-level features, such as shapes and objects. This hierarchical feature extraction allows CNNs to understand the structure and composition of images, enabling them to make accurate predictions. To train a CNN, a large annotated dataset is required. During the training process, the network learns the optimal filter weights and biases that minimize the difference between the predicted outputs and the true labels. This optimization is typically done using stochastic gradient descent (SGD) or its variants, where the gradients of the loss function with respect to the network parameters are computed and used to update the weights. CNNs have achieved remarkable success in various computer vision tasks. They have been used for image classification tasks, such as recognizing objects in images and assigning them to predefined categories. CNNs have also been employed in object detection, where they not only classify objects but also localize their positions in the image. Additionally, CNNs are widely used in image segmentation, which involves dividing the image into meaningful regions or segments.

In conclusion, Convolutional Neural Networks (CNNs) have revolutionized computer vision and image analysis. Their ability to automatically learn and extract relevant features from visual data has made them highly effective in various tasks, such as image classification, object detection, and image segmentation. With advancements in deep learning and the availability of large annotated datasets, CNNs continue to push the boundaries of computer vision and pave the way for new applications and breakthroughs in the field.

Generative models

Generative deep learning models are a class of neural networks that have the ability to generate new data samples that are similar to the training data they were exposed to. These models have garnered significant attention and have shown remarkable potential in various domains, including image synthesis, text generation, and music composition.

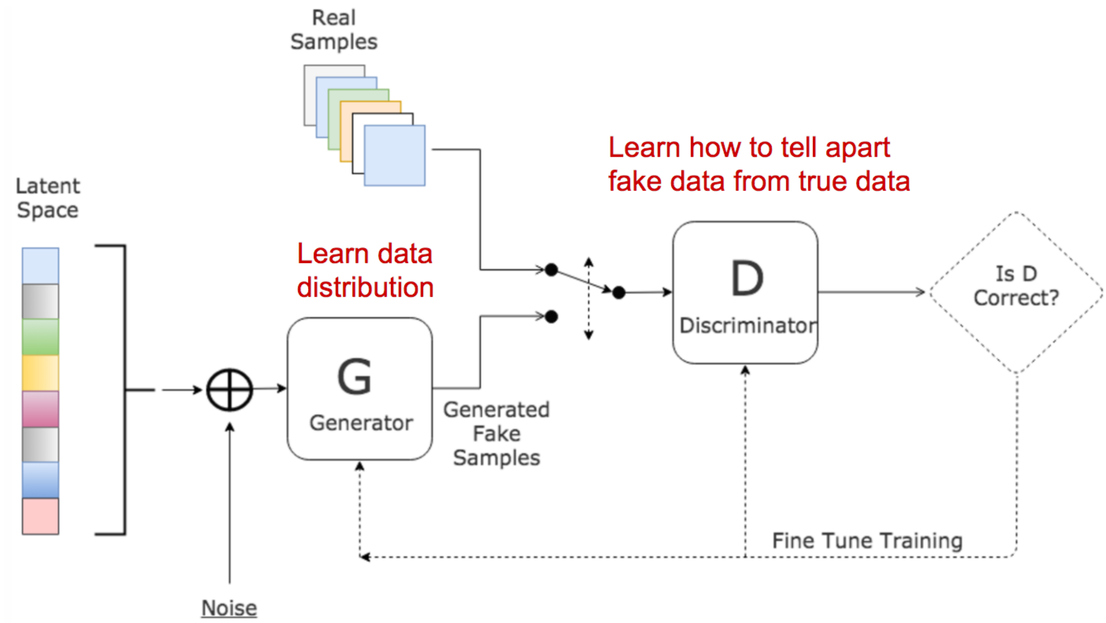

Figure 1c. Illustration of GAN achitecture
(https://lilianweng.github.io/posts/2017-08-20-gan)

One popular type of generative model is the Generative Adversarial Network (GAN). GANs consist of two main components: a generator and a discriminator. The generator takes random noise as input and generates synthetic samples, while the discriminator tries to distinguish between the real and synthetic samples. The two components are trained simultaneously in a competitive manner. The generator aims to produce samples that are indistinguishable from the real data, while the discriminator tries to improve its ability to differentiate between real and synthetic samples. Through this adversarial training process, GANs are able to learn the underlying distribution of the training data and generate new samples that resemble the real data. GANs have been successfully applied in various image synthesis tasks, such as generating realistic-looking images, creating artistic variations, and even modifying images by changing specific attributes. Another widely used generative model is the Variational Autoencoder (VAE). VAEs are based on the idea of learning a latent space representation of the data. The model consists of an encoder that maps the input data to a lower-dimensional latent space and a decoder that reconstructs the original data from the latent space representation. The encoder and decoder are trained together to ensure that the reconstructed samples closely resemble the original input. VAEs are capable of generating new samples by sampling from the learned latent space. By exploring different regions of the latent space, it is possible to generate diverse and novel samples. VAEs have been successfully employed in tasks such as image generation, text generation, and even generating music. In addition to GANs and VAEs, there are other generative models that have gained attention in recent years, such as autoregressive models and flow-based models. Autoregressive models, such as the PixelCNN and PixelRNN, generate data by modeling the conditional probabilities of each element in the data given previous elements. These models have been used for tasks such as image generation and text generation.

Flow-based models, on the other hand, learn a mapping from a simple distribution (e.g., a Gaussian) to a complex data distribution. By applying a series of invertible transformations, these models are able to generate new samples by sampling from the simple distribution and transforming them through the learned transformations. Flow-based models have been successful in image generation tasks and have been used to generate high-quality images. Generative deep learning models have shown great potential not only in generating realistic and high-quality samples but also in aiding creative tasks and data augmentation. They have been used in various applications, such as generating synthetic data for training purposes, generating new artwork, and even enhancing the capabilities of other machine learning models. However, generative models still face challenges in training stability, mode collapse (where the generator produces limited variations of samples), and evaluation metrics. Research in this area is ongoing to address these challenges and further improve the capabilities of generative deep learning models.

In conclusion, generative deep learning models have demonstrated impressive capabilities in generating new and realistic data samples. They have revolutionized image synthesis, text generation, and music composition, among other domains. With continued advancements in deep learning and the development of novel architectures and training techniques, generative models are expected to play an increasingly important role in creative applications and data generation tasks.

Transformer architecture

The Transformer architecture has emerged as a revolutionary approach to sequence modeling in the field of natural language processing (NLP). It was first introduced in the seminal paper ”Attention Is All You Need” by Vaswani et al. in 2017. The Transformer has since become the de-facto standard for various NLP tasks, including machine translation, language generation, and sentiment analysis.

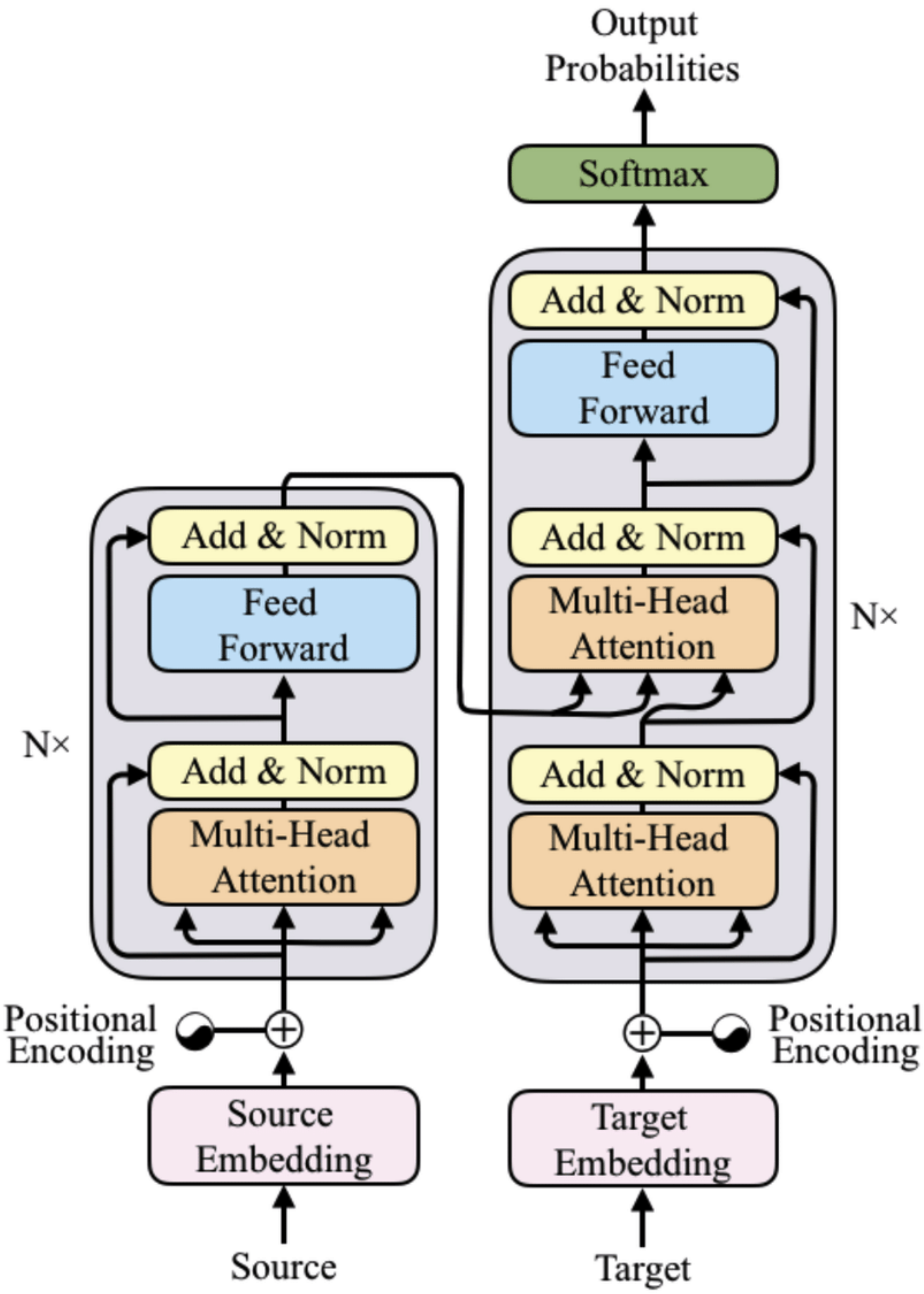

Figure 1d. Illustration of Transformer achitecture
(https://machinelearningmastery.com/the-transformer-model)

The key innovation of the Transformer architecture lies in its attention mechanism. Unlike traditional recurrent neural networks (RNNs) that process sequential data one element at a time, the Transformer is able to capture dependencies across the entire sequence simultaneously. This is achieved through the use of self-attention, also known as scaled dot product attention. In self-attention, each element in the input sequence (e.g., a word in a sentence) is associated with three learned vectors: a query vector, a key vector, and a value vector. The attention mechanism computes a weighted sum of the value vectors, where the weights are determined by the compatibility between the query and key vectors. This allows the model to attend to different parts of the input sequence based on their relevance to the current element being processed. The Transformer architecture consists of multiple layers, each containing a multi-head self-attention mechanism and position-wise feed-forward neural networks. The multi-head attention mechanism allows the model to jointly attend to different positions in the sequence, capturing different types of dependencies. The position-wise feed forward networks apply a non-linear transformation to each position independently, enabling the model to learn complex interactions between elements. One of the advantages of the Transformer architecture is its parallelizability. Unlike RNNs, which process sequences sequentially, the Transformer can process all elements in the sequence in parallel. This makes it highly efficient, especially when dealing with long sequences. Additionally, the attention mechanism allows the model to capture long-range dependencies effectively, which is crucial for tasks such as machine translation. The Transformer also introduced the concept of positional encoding to account for the order of elements in the input sequence. Positional encoding is a set of learned embeddings that are added to the input representation of each element. These embeddings provide the model with information about the position of each element in the sequence, enabling it to capture the sequential nature of the data. The success of the Transformer architecture has been demonstrated across various NLP tasks. In machine translation, the Transformer has achieved state-of-the-art performance on multiple language pairs. It has also been applied to tasks such as language modeling, sentiment analysis, and document classification, consistently outperforming previous models. Furthermore, the Transformer has had a significant impact beyond NLP. Its self-attention mechanism has been adapted to other domains, such as computer vision, where it has been used for tasks like image recognition and object detection. The Transformer's ability to capture global dependencies and model complex interactions has made it a versatile architecture applicable to a wide range of problems. Despite its success, the Transformer architecture does have some limitations. One of the main challenges lies in handling very long sequences, as the self-attention mechanism scales quadratically with the sequence length. Various techniques, such as hierarchical attention and sparse attention, have been proposed to address this issue. Additionally, the Transformer's performance heavily relies on large-scale pre-training and fine tuning, which requires substantial computational resources and labeled data.

In conclusion, the Transformer architecture has revolutionized sequence modeling in NLP. Its attention mechanism enables the model to capture long-range dependencies and process sequences in parallel, leading to impressive results on various tasks. The Transformer's success has spurred further research and advancements in the field, pushing the boundaries of what is possible in natural language processing and beyond.

3. Challenges

Data requirements

Deep learning models need to be trained on large amounts of data in order to learn complex patterns. This can be a challenge for a number of reasons. Collecting and labeling large amounts of data for deep learning models can be expensive and time-consuming. Additionally, some tasks, such as medical diagnosis and fraud detection, require sensitive data that can be challenging to collect and share. Furthermore, certain tasks, like natural language processing and machine translation, necessitate substantial amounts of text data, which may not be readily available for all languages. In addition to the challenges listed above, deep learning models are also sensitive to the quality of the training data. If the training data is noisy or biased, the model will learn to produce noisy or biased predictions.

There are a number of ways to address the challenges of data requirements in deep learning. Transfer learning is a technique where a pre-trained deep learning model is used as a starting point for training a new model on a different task. This approach can reduce the amount of data required to train a new model by leveraging knowledge learned from pre-trained models on large datasets, allowing for the reuse of learned features and representations. Data augmentation is a technique where new training data is created by applying transformations to existing training data. This can increase the size and diversity of the training data without having to collect new data. Despite the challenges, deep learning can solve a wide variety of problems. By using the techniques described above, it is possible to train deep learning models even with limited data.

Researchers are working on developing new techniques to address the challenges of data requirements in deep learning. One promising area of research is data augmentation, which involves generating new training data from existing data. Another promising area of research is transfer learning, which involves using pre-trained models to train new models on new tasks.

Interpretability

Interpretability in large language models (LLMs) like GPT-3 is a critical aspect that refers to our ability to understand and interpret how these models make their predictions or decisions. It centers around the transparency, comprehensibility, and explainability of artificial intelligence algorithms. Firstly, transparency denotes the level of access an observer has to the inner workings of the model. For LLMs, this is challenging due to their ‘black-box’ structure, where thousands to millions of parameters are involved in generating outputs. Comprehensibility relates to the simplicity of understanding the model’s decision-making process. For instance, linear regression models are highly comprehensible as they use a straightforward weighted combination of input features. However, LLMs involve complex nonlinear dynamics across high-dimensional spaces, making them harder to comprehend. Explainability pertains to providing understandable reasons for the decisions made by the model. An interpretable model can provide human-understandable explanations for why it generated a certain output given a specific input, which is particularly important in high-stakes domains like healthcare or finance. Currently, understanding what goes on inside LLMs is a significant challenge due to their complexity and lack of inherent interpretability. These models are trained on vast amounts of data and create layers of hidden representations used to generate predictions. The relationships between these representations and final outputs often appear opaque, even to the engineers who design and train these models. Various techniques have been developed to improve the interpretability of such models. Feature importance analysis, for example, can highlight which inputs the model considers most vital for its predictions. Visualization tools can show activity within layers of the model during processing, while natural language explanations can describe the rationale behind decisions in human language. However, these methods often fall short of providing full interpretability. They offer glimpses into the model's operations but do not fully reveal the intricate web of computations and transformations happening within. In fields where decisions have significant consequences, understanding why a model made a certain choice can be crucial. It allows for better error analysis and debugging, promotes trust, ensures fairness by detecting biases in predictions, and aids in regulatory compliance. In conclusion, interpretability in LLMs is a complex subject, essential for the ethical and effective deployment of these models. It's a topic of ongoing research with the aim of making AI systems more transparent, comprehensible, and explainable, thereby aligning them more closely with human values and standards.

One primary difficulty lies in how LLMs make decisions. Unlike traditional rule-based systems, where logic is clearly defined and traceable, LLMs are based on deep learning techniques that involve multiple layers of intricate neural networks. These networks generate outputs through non-linear computations, making it challenging to understand or predict their behavior. This opacity, often referred to as the ”black box” problem, hinders the interpretability of LLMs. Another challenge arises due to the sheer complexity and size of these models. LLMs, like GPT-3, with billions or even trillions of parameters, are trained on vast quantities of data. The exact contribution of each parameter to the output is hard to ascertain, further complicating interpretability. Moreover, since these models learn from massive datasets, they might also internalize and reproduce any biases present in the training data, leading to potential ethical concerns. A third challenge revolves around the unpredictable nature of LLM responses. Without an understanding of how the model generates its outputs, there's a risk of it providing inappropriate or harmful information. For instance, an LLM may inadvertently generate false or misleading content if it misinterprets the nuances of a given context, such as sarcasm or cultural sensitivities. Understanding why an LLM rejects certain inputs while accepting others can be equally perplexing. For example, when asked similar questions with minor wording changes, LLMs may provide different answers, reflecting the sensitivity to the input phrasing. However, without clear interpretability, it becomes tricky to explain why such variations occur. Efforts towards enhancing interpretability include techniques like attention maps, feature importance, and layer-wise relevance propagation. While helpful, these methods still lack robustness and may not provide comprehensive insights into the mechanisms of complex LLMs. In addition, while explainable AI (XAI) techniques aim to increase model transparency, they often involve a trade-off between performance and interpretability. Simpler models are easier to interpret but may not provide the desired accuracy or generalization. Conversely, more complex models offer higher performance but at the cost of lower interpretability. Addressing interpretability challenges requires advances in both theoretical understanding and practical tools for probing LLMs. This involves finding a balance between creating powerful models that can effectively handle sophisticated tasks and ensuring they remain understandable and controllable. Ultimately, improving the interpretability of LLMs is crucial for fostering trust and ensuring their reliable integration into various applications and systems.

There are, however, several techniques and strategies that researchers are exploring to improve interpretability in these models.

• Attention Visualization: Attention mechanisms in transformer-based models like GPT-3 allow the model to ”focus” on different parts of the input when producing an output. By visualizing these attention weights, we can gain some insight into which parts of the input the model considers most relevant. However, the interpretation from these visualizations is not always straightforward due to complex interactions between different layers and heads in the model.

• Feature Importance Analysis: Techniques like LIME (Local Interpretable Model-Agnostic Explanations) or SHAP (Shapley Additive exPlanations) can highlight which parts of the input contribute most strongly to the prediction. These techniques perturb the input and observe changes in the output, which provides local interpretability around a particular prediction.

• Probing Tasks: Probing tasks are designed to investigate whether certain types of information are encoded in the model’s representations. A common approach is to train a simple classifier, or ’probe’, to predict a specific characteristic using the model’s intermediate representations as input. This technique helps in understanding what knowledge the model captures.

• Counterfactual Explanation: Another way of interpreting AI decisions is by generating counterfactual explanations. A counterfactual explanation describes the closest possible world where the model’s decision would change, providing insights into what aspects of the input the model is sensitive to.

• Model Simplification: Some research focuses on simplifying complex LLMs into smaller, more interpretable models that approximate the original model’s behavior. While these simplified models may lose some accuracy, they can provide a better understanding of how the original model works.

• Causal Inference: Causal inference techniques attempt to understand the cause-effect relationship between input features and output predictions. This can help illuminate how changes in specific parts of the input might lead to changes in the output.

• Transparency through Training: Transparency can also be improved by incorporating interpretability objectives during training. Techniques such as Rule Extraction From Trained Neural Networks (RELU) or introducing interpretability loss terms promote a model that is more interpretable without compromising performance.

In conclusion, ensuring interpretability in LLMs is not only key to understanding their decision-making process but also crucial for establishing trust, especially in high-stakes applications. While none of these techniques provide a complete solution, they each contribute pieces to the larger interpretability puzzle. As we continue to refine our methods and develop new ones, our ability to peer inside these powerful tools will only improve.

Cost of training

The cost of training a deep learning model can be divided into the following components. Collecting and labeling large amounts of data can be expensive and time-consuming. Deep learning models require access to powerful computational resources, such as GPUs, to train effectively. Developing and tuning deep learning models requires expertise and experience.

There are a number of ways to reduce the cost of training deep learning models. Transfer learning is a technique where a pre-trained deep learning model is used as a starting point for training a new model on a different task. This can reduce the amount of data and computational resources required to train the new model. Data augmentation is a technique where new training data is created by applying transformations to existing training data. This can increase the size and diversity of the training data without having to collect new data. There are a number of efficient training algorithms that can be used to train deep learning models. These algorithms can reduce the amount of time and computational resources required to train the model (e.g. distributed training).

Despite the challenges, deep learning is a powerful tool that can be used to solve a wide variety of problems. By using the techniques described above, it is possible to reduce the cost of training deep learning models. Researchers are working on developing new techniques to reduce the cost of training deep learning models. One promising area of research is distributed training, which involves training a deep learning model on multiple machines simultaneously. This can help to reduce the cost of computational resources.

Bias

Bias in LLMs refers to the tendency of these models to favor certain ideas, perspectives, or groups over others in a way that can be unjust or unrepresentative.

This bias typically originates from two main sources: the training data and the model architecture. The first source of bias, the training data, includes all the text the model learns from. If this data contains biases—whether explicit or implicit—the model will likely learn and reproduce them. For example, if a model were trained primarily on English language data, it might struggle to generate accurate or nuanced responses in other languages, thus demonstrating a language bias. Furthermore, if the training data includes stereotyped or prejudiced views, these too could be echoed by the model. The second source of bias lies within the model architecture itself, specifically in its design decisions—how it processes input, how it weighs different pieces of information, etc. For instance, some models utilize attention mechanisms that prioritize certain parts of an input over others. Depending on how they're implemented, such mechanisms could introduce bias by systematically over or under-representing certain inputs. Bias in LLMs has significant implications. It can lead to outputs that reinforce harmful stereotypes, discriminate against certain groups, and misrepresent reality. Moreover, when users interact with biased LLMs, they may unknowingly internalize the biases present in the model's responses, exacerbating societal biases further. Addressing bias in LLMs is a complex but essential task. It begins with efforts to diversify and balance the training data and continues with careful consideration of model architecture. In addition, techniques like bias audits, which involve systematic testing for discriminatory behavior, are critical. OpenAI, for instance, is committed to reducing both glaring and subtle biases in its AI models' responses through research and engineering methods. To sum up, bias in LLMs is a pervasive issue originating from both training data and model design. It can lead to problematic outputs that echo societal prejudices and stereotypes, underscoring the need for ongoing efforts in bias identification, mitigation, and prevention. As LLMs become more integrated into society, addressing these biases will only grow in importance.

Bias in LLMs refers to the tendency of these models to favor certain perspectives, ideologies or demographics over others in their output. This bias often reflects imbalances in the data sets used to train them. Since these models learn from large volumes of text data, any biases present in these texts can be absorbed and reproduced by the models. This can lead to stereotypes, discrimination, or misrepresentations - all forms of bias that can have serious, real-world impacts. The first major challenge in dealing with bias in LLMs is detection. It's difficult to quantify or measure bias in a model's output because it's not always obvious, such as instances of subtle prejudice or systemic bias. The sheer complexity and opacity of LLMs pose further difficulties in pinpointing specific areas where bias originates. Without understanding the source of bias, addressing it effectively becomes complicated. Secondly, there's the challenge of defining what constitutes 'bias'. Opinions can greatly vary on what is considered biased or unbiased content. Striking the right balance without infringing upon freedom of expression or promoting censorship represents a tricky dilemma. As LLMs operate globally, harmonizing diverse cultural norms and values into a universally acceptable 'unbiased' model is nearly impossible. A third challenge lies in the mitigation or reduction of bias. Whilst retraining models with more balanced data or using techniques like adversarial training could help, these are time-consuming and technically arduous. Furthermore, bias is not always straightforward; indirect bias, where seemingly neutral inputs lead to biased outputs, might persist even after direct bias is addressed. Lastly, ethical considerations pose a significant challenge. Deciding who gets to define biases, who has the authority to 'correct' these biases, and what the consequences might be, are all deeply ethical questions without clear-cut answers. This can lead to power imbalances or misuse of these models. In conclusion, bias in LLMs is a complex issue that requires multidisciplinary efforts to tackle. Research continues to explore techniques for better bias detection, definition, and mitigation, as well as address the associated ethical dilemmas. While it may not be possible to completely eradicate bias from LLMs, it's crucial to strive towards minimizing its harmful impacts through transparency, accountability, and continuous dialogue.

Dealing with bias in Large Language Models (LLMs) like GPT-3 is a multifaceted challenge, requiring a range of techniques to ensure that these models provide fair, unbiased outputs. Bias can emerge due to the skewed nature of the training data or the model's learning mechanisms. One critical technique is "bias mitigation during pre-training". LLMs are trained on vast amounts of text data from the internet which often include inherent biases. To mitigate this, developers can curate the training datasets wherever possible to minimize prejudiced content. Such efforts can be resource-intensive; however, they greatly contribute to reducing the model's exposure to biased data. Another technique is "bias mitigation during fine-tuning." Fine-tuning is a process that allows models to learn specific tasks or adapt to certain domains after pre-training. Developers can use carefully designed data sets during this stage to 'correct' some of the biases learned during pre-training. They can also implement fairness constraints or penalties for biased predictions to further reduce inherent prejudices. "Transparent reporting and evaluation" is an essential part of managing bias. Researchers should transparently communicate about potential biases in their models. Evaluation metrics crafted explicitly for bias detection can also be of great value. These measurements can focus on various aspects, such as studying the model's behavior across different demographic groups and assessing whether the model is perpetuating harmful stereotypes. The application of "external audits and third-party evaluations" has proven to be fruitful. Independent audits of AI systems can help identify hidden biases and provide recommendations for improvement. External audits help counteract confirmation bias and improve the objectivity of the analysis. "User customization and control over system behavior" can empower users to modify a system's output according to their ethical values within broad societal limits. This approach recognizes the plurality of ethical perspectives and respects user autonomy while ensuring the system does not enable malicious uses. "Public input on defaults and hard bounds" involves gathering collective input on the system behavior and defining hard boundaries. Since AI impacts a wide population, it is crucial for decisions about system behavior to incorporate perspectives from those who use or are affected by systems. Lastly, "continuous learning and adaptation" is important. As cultural norms and societal values evolve, so too should our understanding of bias, fairness, and fairness metrics. Therefore, it is essential that techniques for dealing with bias in LLMs incorporate an element of ongoing learning and evolution. In sum, dealing with bias in LLMs requires a combination of technical methods and human oversight, involving pre-training and fine-tuning interventions, transparent reporting, external reviews, user customization, public inputs, and continuous learning strategies with the aim of improving fairness and reducing harm.

Controllability

Controllability is a term that encapsulates the power to guide or direct a system's behavior in order to achieve specific results. The origin of this term can be traced back to control theory, a branch of engineering that deals with the behavior of dynamical systems, where it means the ability to move a system from any initial state to any desired final state within a finite time interval, typically with the use of input control functions. In the realm of deep learning models, the concept of controllability takes on an added layer of importance. It refers to the degree to which an operator, often a human user or another system, can influence and tailor the outcomes produced by a model. This influence can be exerted at various stages and levels, including during the training phase, the data-feeding process, and the design of the model architecture. Deep learning models, as part of machine learning, are inherently designed to learn patterns from massive amounts of data and generate predictions or decisions based on those patterns. Therefore, the ability to control them implies having mechanisms to guide these predictions toward ethically sound, practically useful, and situationally appropriate results. The need for controllability in deep learning is amplified due to a variety of reasons. First are applications where human intervention is necessary. Although deep learning models have been successful in automating several tasks, there are areas where they operate in tandem with humans, requiring a certain level of control. For example, in healthcare, a deep learning model might assist doctors in diagnosing diseases by analyzing medical images. However, a doctor must have the ability to override or adjust the model's suggestions based on their expertise and the patient's unique context. Ethical considerations also necessitate control over deep learning models. As artificial intelligence continues to permeate all spheres of life, issues related to fairness, transparency, and privacy have emerged. Ensuring that AI systems do not inadvertently perpetuate bias or invade privacy requires the ability to steer the systems' behavior. Specific constraints, such as legal or regulatory requirements, can also demand controllability in deep learning. For instance, certain sectors such as finance or healthcare are bound by stringent regulations related to data handling and decision-making processes. In these cases, controllability implies having means to ensure that the operations of deep learning models comply with these norms. In conclusion, controllability in the context of deep learning systems is more than a mere technical term. It represents an essential quality that aligns the technology with human values, professional judgment, ethical standards, and societal rules. The ongoing research in explainable AI, fair AI, and human-in-the-loop AI all point towards an increased emphasis on enhancing the controllability of deep learning models for their responsible use.

Advancements in artificial intelligence, particularly within the realm of deep learning, have led to remarkable strides in diverse sectors such as healthcare, finance, and autonomous vehicles. However, along with these opportunities come significant challenges, especially concerning model control. Firstly, explainability, or ”interpretability,” is a critical issue. Deep learning models are often termed ”black boxes" because it is typically difficult to understand how they arrive at their decisions. Some models, like convolutional neural networks (CNNs), take thousands or millions of inputs and apply many layers of computation. Determining which inputs significantly influenced the output is challenging because the decision making process gets lost in complex, interconnected mathematical functions. This lack of transparency hinders trust and broad adoption, especially in sectors where explainability is legally required or ethically imperative, like healthcare or criminal justice. Secondly, there’s the challenge of bias and fairness. Deep learning models learn from data, and if the training data contains biases, the model will likely reproduce those biases. For instance, facial recognition systems have been shown to perform unequally across different ethnic groups due to biased training sets. This can lead to discriminatory outcomes, making it crucial to develop techniques for identifying and mitigating bias. A third major issue relates to adversarial attacks. These are designed to mislead models through malicious input, leading to incorrect outputs. Such attacks pose serious security risks, particularly in sensitive applications like cybersecurity or autonomous vehicles. Despite numerous defenses proposed, creating models robust against all forms of adversarial manipulations remains an open problem. Privacy concerns constitute another significant challenge. Models trained on sensitive data can inadvertently reveal information about individuals. Differential privacy techniques are being developed to mitigate this risk, but striking a balance between utility and privacy protection is intricate. Lastly, the cost and environmental impact associated with training large scale deep learning models cannot be ignored. The compute resources needed for state-of-the-art models have significant carbon footprints and are often beyond the reach of smaller organizations or researchers, leading to a centralization of AI power. In conclusion, controlling deep learning models is a multifaceted challenge involving technical, ethical, and societal aspects. It is imperative that as we advance in our capabilities with these powerful tools, we also focus on refining control mechanisms to ensure their transparent, fair, secure, privacy-preserving, and sustainable use. The future impact of AI on society will largely depend on how effectively we manage these challenges.

Language Learning Models (LLMs) are increasingly being adopted in various industries due to their impressive ability to mimic human-like conversation and understanding. However, controlling these models to yield predictable, safe, and user specific outcomes remains a critical concern for many developers and users. One fundamental technique is pre-training and fine-tuning. Initially, models are pre-trained on a vast corpus of text data, enabling them to understand the structure of the language. Then, they are fine-tuned on specific datasets related to the task at hand or incorporating user preferences. This two-step process ensures models align with general linguistic patterns while also adapting to specific requirements. However, this alone may not be enough. An emerging practice to control LLMs' output is reinforcement learning from human feedback (RLHF). In this technique, an initial model is trained using comparison data – pairs of model outputs ranked by quality. A reward model is then trained to predict those rankings, which further trains the model via Proximal Policy Optimization. Through several iterations, the model can learn more nuanced behavior from intricate feedback. Another approach is the use of prompts. Since LLMs respond based on the pattern of input they gain, fine-tuning the prompt structure can effectively guide the model's responses. For example, detailed prompts that provide context or questions formulated in a certain way can elicit more controlled and desired responses. Moreover, specifying external constraints during interaction with the model offers real-time control. These could include system-level constraints like banning output of certain types of content, setting context windows, or user-defined instructions embedded within the input. Despite these techniques, LLMs may still produce outputs with biases or inaccuracies inherent in their training data or fail to reject inappropriate requests. To address this, developers are considering controllable knobs or dials users can adjust to influence the model's behavior, such as the degree of creativity or verbosity. Further research into AI safety is exploring methods like rule-based rewards and Constitutional AI, where ethical guidelines or specific rules are embedded within the model's architecture itself. This can control LLMs by limiting them to function within predetermined boundaries. As we continue to use and develop LLMs, ensuring they understand and respect user values becomes paramount. Techniques such as value learning aim to make LLMs learn and align with the values of their users while maintaining safety precautions against malicious use. Ultimately, controlling LLMs is a multifaceted challenge that requires a combination of these techniques. The goal is to create a balanced system that respects user autonomy and ensures safety, while continually learning and improving from feedback and interaction. As this field advances, researchers need to keep iterating on these techniques to ensure the safe and beneficial use of LLMs.

Controlling deep learning models, especially LLMs, is a challenging yet critical area of research. While these models offer tremendous capabilities, achieving controllability is essential to ensure their safe and responsible use. Researchers are actively exploring various techniques, including fine tuning, guided generation, rule-based constraints, and human-in-the-loop approaches, to enhance the controllability of LLMs. By addressing these challenges, we can unlock the full potential of deep learning models while maintaining control over their outputs.

Hallucination

Language Learning Models (LLMs) like GPT-3 by OpenAI use machine learning to understand and generate human language. The concept of ”hallucination” in LLMs refers to instances where the model generates outputs that seem plausible but are actually not factual or grounded in reality. This usually happens when the model makes guesses or assumptions based on patterns it has learned from its training data, leading to incorrect or completely fabricated information. Hallucination is a direct result of the fundamental architecture of these models. They do not have access to real time facts or real-world context beyond what they were trained on. If an input triggers a learned pattern associated with false information, the model might generate outputs that reflect this falsehood, hence ’hallucinating’. For instance, an LLM might generate a sentence claiming that a particular historical figure is still alive, despite them having passed away years ago. This could be because the model was trained with data up until a point when the individual was still alive, and it hasn’t been updated with new data since their passing. Another form of hallucination can occur when the model generates outputs that are internally consistent and grammatically correct but are entirely fictitious. For example, it might describe a fictional scientific theory or event that sounds plausible but doesn’t exist in reality. These hallucinations are often more difficult to spot because they might not directly contradict known facts. The concept of hallucination in LLMs poses significant challenges for the use of these systems, especially in applications where accurate, fact-based responses are crucial, such as news generation, educational tools, or legal advice. It underscores the importance of rigorous validation, verification, and continual updating of training data to ensure the accuracy of generated outputs. Moreover, it also highlights the need for developing more sophisticated mechanisms within LLMs to reduce hallucination. This could include more advanced forms of fact-checking capabilities, the incorporation of real time data updates, or the development of systems to allow for external fact-checking against a reliable database. In conclusion, hallucination in LLMs is a complex issue that arises from these models’ inherent limitations. It’s an area of active research and presents both technical and ethical challenges for AI developers and users alike.

Language Learning Models (LLMs), like the ones developed by OpenAI, represent some of the most advanced manifestations of AI. However, they aren’t free from challenges, one of the prominent ones being dealing with hallucinations or generating information that isn’t based on the input or training data. Hallucination in LLMs primarily refers to situations where models generate output that may sound plausible but are incorrect or unverified. This can range from minor alterations to completely fabricated facts. Hallucination poses a significant challenge because it can mislead users and erode trust in the reliability of these systems. It also raises concerns regarding misinformation, since unchecked hallucinated content could spread false narratives. Several factors contribute to hallucination in LLMs. Training data is arguably the most critical element. If the training data includes incorrect or misleading information, the model will learn and potentially replicate those inaccuracies. Also, if training data lacks representation of certain types of information, the model might fill in gaps with hallucinated details. Furthermore, deep learning models like LLMs work largely through pattern recognition rather than understanding, which further exacerbates the hallucination problem. The ambiguous nature of natural language can also lead to hallucinations. Language is filled with nuances, context-specific meanings, and interpretive elements, which can be challenging for LLMs to grasp fully. When presented with ambiguous prompts, these models may default to producing responses that seem plausible but are not grounded in facts. Dealing with hallucinations in LLMs calls for novel solutions. One approach is improving the quality and diversity of training data to ensure the model has a wide and accurate base to learn from. Ensuring the dataset represents all necessary aspects of the real world, especially rare events, can help reduce hallucinations. Another approach is leveraging reinforcement learning from human feedback. Users can actively point out when the model hallucinates, informing updates to its training. However, this can be challenging due to the large amount of data required and potential biases in user feedback. Additionally, researchers are exploring techniques that incorporate explicit world knowledge into these systems. Known as knowledge graphs, they provide a structured format for facts that could serve as a ”fact-checker” for LLMs. Designing safety mitigations, such as allowing users to flag or downvote incorrect outputs, or integrating internal checks within the model to cross-verify the information it generates, can also be beneficial. While we’ve made strides in developing LLMs, dealing with hallucination remains a significant challenge. It demands rigorous research efforts and better training methodologies to ensure the trustworthy and responsible deployment of these advanced AI systems.

Controlling hallucination, or the generation of incorrect and unfounded information, is a crucial task for large language models (LLMs) like OpenAI’s GPT-3. Striking a balance between creativity and accuracy in an LLM’s responses can be complex. Here are some techniques used to control hallucination.

• Fine-tuning: This process involves training the model on a specific dataset after its initial training. The data used for fine-tuning is typically narrow and selected to reflect the desired output. For instance, if you want the model to generate accurate scientific information, you might fine-tune it with a dataset comprising scientific literature.

• Reinforcement Learning from Human Feedback (RLHF): In RLHF, models learn to generate better content by interacting with humans. They generate a response, receive feedback on its quality, and use this information to adjust future outputs. Through this, models learn to avoid hallucinating information that humans don’t validate.

• Controlling Generation Temperature: Models like GPT-3 use a parameter called ’temperature’ during the text generation process. Higher temperature values lead to more random outputs, while lower values make the output deterministic. By tuning the temperature, one can control the degree of creativity and potential hallucination in the model’s responses.

• Prompt Engineering: The design of the input prompt can significantly influence the model’s output. By asking clear, direct questions and providing necessary context, you can guide the model towards producing accurate, non-hallucinated information. • Using External Fact Checkers: An external fact-checker, another AI algorithm that verifies the claims made by the LLM, could be employed. If the fact-checker determines that the LLM is generating false information, it can signal the LLM to modify its generation process.

• Limiting Output Length: Shorter responses often contain less room for error or fabrication compared to longer ones. By limiting the output length, the risk of generating hallucinated information can be reduced.

• Systematic Evaluation and Iterative Refinement: Regularly evaluating the model's performance and iteratively refining it based on these evaluations can help control hallucinations. This might involve both automated evaluation methods and human evaluations.

In conclusion, controlling hallucination in LLMs is a multifaceted problem that requires a combination of techniques. These approaches need to be applied strategically and often in tandem to effectively reduce the occurrence of hallucination while maintaining the model's ability to generate useful, creative responses.

4. Language model architecture

Language model architectures play a crucial role in various natural language processing (NLP) tasks. Three common architectures are encoder, encoder-decoder, and decoder. Each architecture has its specific use cases, advantages, and disadvantages.

Differences between Self-attention and Cross-attention

Self-attention, a fundamental component of the transformer architecture, has revolutionized natural language processing (NLP) by enabling models to capture long-range dependencies between words and comprehend the overall meaning of a sentence. This mechanism operates by allowing a model to attend to different positions within the same input sequence, establishing connections between words that may be far apart. To understand how self-attention works, consider a sentence like "The quick brown fox jumps over the lazy dog." When the model processes this sentence using self-attention, it can learn to associate the words "quick" and "fox," even though they are separated by three other words. This ability to capture long-range dependencies is crucial for tasks like machine translation, where the model needs to understand the relationships between words across different languages.

While self-attention focuses on relationships within a single sequence, cross-attention expands the scope by enabling a model to attend to words in two different input sequences. This mechanism is particularly useful in encoder-decoder architectures, where the encoder processes the input sequence and the decoder generates the output sequence. Cross-attention allows the decoder to incorporate information from the encoder's representation of the input sequence, ensuring that the generated output is relevant and consistent with the input. For example, in machine translation, cross-attention plays a critical role. The decoder, when generating the translated sentence, can attend to the encoder's representation of the source language sentence. This allows the decoder to consider the grammatical structure, vocabulary choices, and overall meaning of the source sentence, leading to more accurate and natural-sounding translations.

The primary distinction between self-attention and cross-attention lies in the source of information they utilize. Self-attention draws from a single input sequence, enabling it to capture long-range dependencies within that sequence. Cross-attention, on the other hand, bridges the gap between two different input sequences, allowing the model to compare and contrast them and understand their relationship.

Encoder Architecture

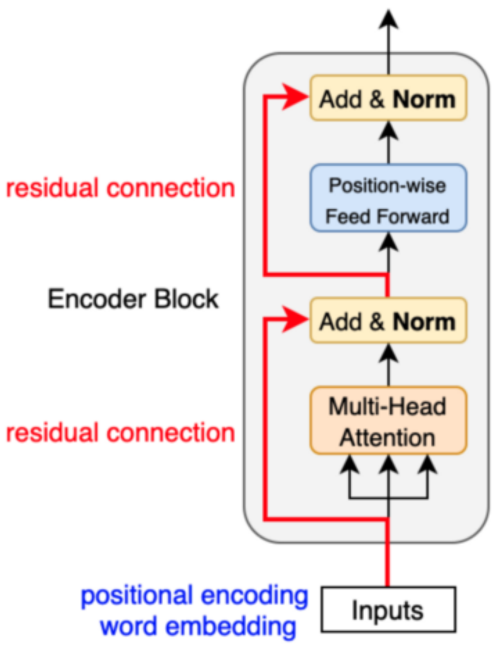

Figure 2a. Illustration of Encoder achitecture (https://kikaben.com/transformers-encoder-decoder)

An encoder-only architecture consists of a single encoder stack that processes an input sequence and produces contextualized representations of each token in the sequence. These representations capture the meaning and relationships between the tokens in the input, allowing the model to perform various tasks such as classification, question answering, and summarization. The encoder's self-attention mechanism allows each token to attend to all other tokens in the input sequence, enabling it to extract global information and understand the context of each token. The encoder also includes feed-forward layers that add non-linearity to the representations.

Encoder-only models are particularly effective for tasks that require understanding and encoding the meaning of an input sequence, such as: (1) Natural

Language Understanding (NLU): Tasks like sentiment analysis, topic classification, and text summarization involve understanding the meaning of an input text and extracting relevant information. (2) Question Answering (QA): Encoder-only models can be used to answer questions about a given text passage by first encoding the passage and then using the encoded representations to answer the question.

Encoder-Decoder Architecture

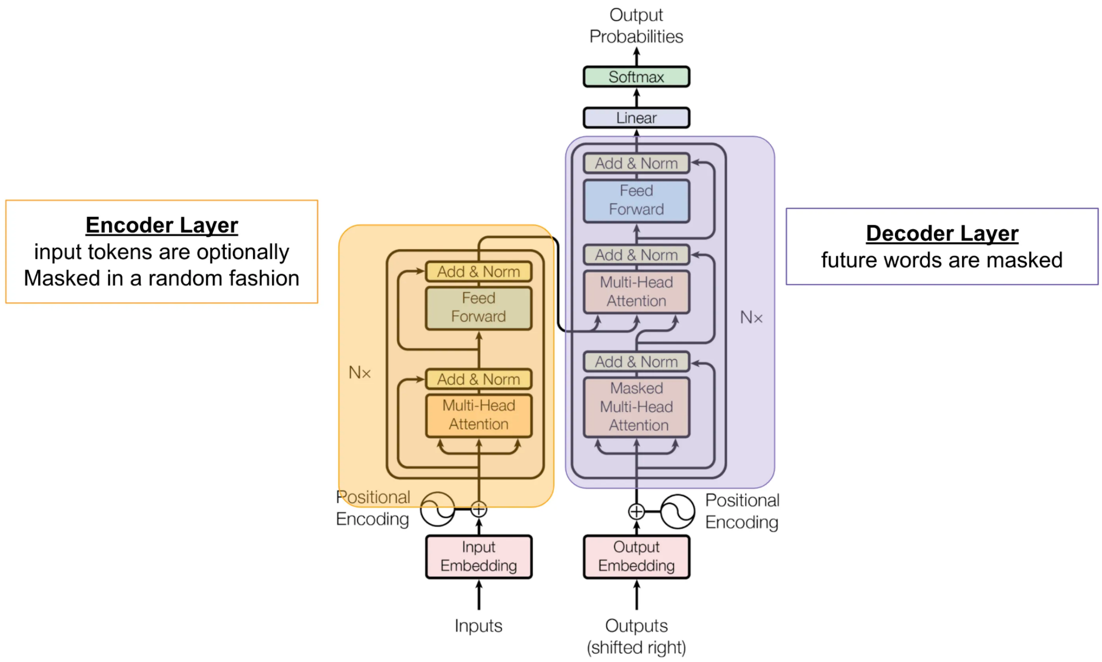

Figure 2b. Illustration of Encoder-Decoder achitecture (https://aiml.com/explain-the-transformer-architecture)

The encoder-decoder architecture is the most common transformer architecture, consisting of an encoder stack followed by a decoder stack. The encoder processes the input sequence and produces contextualized representations, while the decoder generates an output sequence based on the encoder's representations and its own internal state. The decoder's self-attention mechanism allows each token in the output sequence to attend to all other tokens in the output sequence, ensuring that the output is coherent and consistent. The decoder also includes feed-forward layers and an attention mechanism that allows it to attend to the encoder's representations, allowing it to generate an output that is relevant to the input.

Encoder-decoder models are particularly effective for tasks that require generating an output sequence based on an input sequence, such as: (1) Machine Translation (MT): The encoder encodes the source language sentence, and the decoder generates the corresponding translation in the target language. (2) Text Summarization: The encoder encodes the input text, and the decoder generates a shorter summary that captures the main points of the input. (3) Sequence-to-Sequence

Learning (Seq2Seq): A broad category of tasks where the model learns to transform one sequence of data into another, such as text-to-speech or speech-to-text translation.

Decoder Architecture

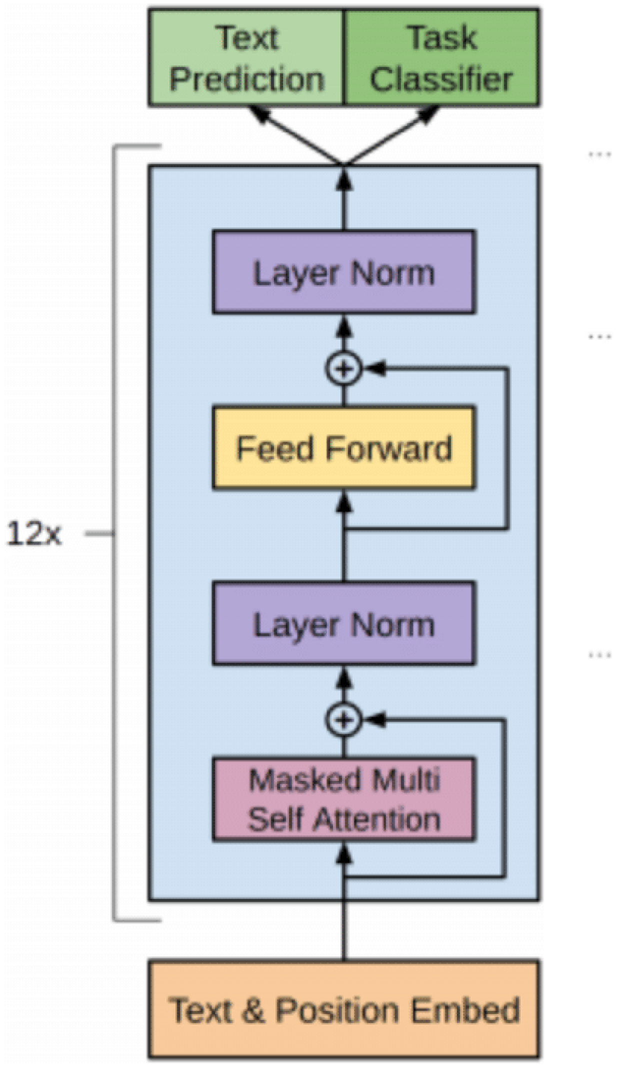

Figure 2c. Illustration of Decoder achitecture (https://www.researchgate.net/figure/Decoder-Only-Architecture-used-by-GPT-2_fig1_349521999)

The decoder-only architecture consists of a single decoder stack that generates an output sequence based on a prefix or an initial context. This architecture is particularly useful for tasks that require generating creative text formats, such as: (1) Creative Writing: The decoder can generate poems, code, scripts, musical pieces, email, letters, etc., based on a given prompt or style. (2) Dialogue Generation: The decoder can generate responses to prompts or continue conversations, mimicking human conversational patterns. (3) Text Generation: The decoder can generate new text based on a given context or prompt, such as generating product descriptions or creative text formats.

The decoder-only architecture is gaining popularity due to its simplicity and effectiveness in generating creative and coherent text. However, it is important to note that decoder-only models may not be as effective for tasks that require understanding and encoding the meaning of an input sequence.

In practice, these architectures are often combined and adapted to suit specific NLP tasks. For example, the trans former architecture, which is widely used in modern language models, combines the encoder and decoder components to enable self-attention and capture long-range dependencies.

5. *State of the art*

ChatGPT

The Chatbot Generalized Pre-training Transformer (Chat-GPT) model is a groundbreaking development in the field of artificial intelligence (AI), specifically within natural language processing (NLP). Developed by OpenAI, GPT is a transformer-based language model that employs machine learning techniques to generate human-like text. The underlying technology for Chat-GPT is originally based on the transformer architecture used in the GPT model series, including GPT-1, GPT-2, and the most recent, GPT-3. Each iteration has expanded on prior models, vastly increasing the scale and introducing novel training techniques. The transformer architecture allows the model to handle long-range dependencies in text very efficiently, something earlier architectures struggled with significantly. Chat-GPT leverages unsupervised learning during its initial pre-training phase. It learns to predict the next word in a sentence using a vast amount of internet text data. This process allows the model to acquire an understanding of grammar rules, facts about the world, reasoning abilities, as well as various biases present in the data it was trained on. After pre-training, fine-tuning follows, where the model is further trained on a dataset curated by humans, which contains demonstrations and comparisons to guide its responses. This two-step process of pretraining and fine-tuning is crucial to developing a chatbot like Chat-GPT that can create contextually relevant, nuanced, and coherent text beyond simply generating disconnected sentences. One key aspect of GPT-based models is their generality. Instead of creating separate models for different tasks such as translation, summarization, or sentiment analysis, GPT models are capable of performing these diverse tasks without task-specific modifications. The user merely needs to provide prompts to instruct the model what task to perform, a feature made possible by instruction following capabilities developed through fine-tuning. While having impressive capabilities, it's essential to note potential issues with Chat-GPT. Its reliance on training data means it can sometimes generate inappropriate content or manifest biases present in the data. In response, OpenAI has focused on improving default behavior, enabling user customization within broad boundaries, and soliciting public input on defaults and hard bounds to prevent misuse. Looking forward, advancements in AI models like Chat-GPT continue to push the boundary of what's possible in machine-human interactions. As we refine these models, their potential applications across industries - from customer service to education, entertainment, and beyond - continue to expand. Despite the challenges, GPT-based models such as Chat-GPT represent a significant leap forward in NLP and AI at large. They encapsulate the increasingly sophisticated interplay between technological innovation and societal impact, prompting not only an ongoing effort to improve technical aspects but also a deeper exploration of ethical and policy considerations.

FlanT5

Flan-T5, a novel machine learning model introduced after 2021, is an advanced achievement in the field of Natural Language Processing (NLP). Combining the strength of two powerful models - Flan and T5 (Text-to-Text Transfer Transformer) from Google's research team - Flan-T5 brings new insights to NLP tasks by leveraging a broader context and generating more accurate results. The core principle behind T5 was transmuting every NLP problem into a text-to-text task. It includes tasks like translation (translating Spanish to English), summarization (transforming long paragraphs into short summaries), sentiment analysis (converting sentences into positive or negative sentiments), and much more. The T5 model's success lies in its simplicity and versatility, allowing it to be used across various NLP tasks without major modifications. On the other hand, the Flan model represents an upgrade over traditional transformer models, offering a significant leap in processing efficiency without compromising on performance. By employing Fast Attention via positive Orthogonal Random features (FAVOR+), Flan manages to sidestep the issues confronted by typical attention mechanisms like scaling difficulties associated with sequence length. This ability makes it particularly suited for handling longer text sequences. By fusing these two models' principles, Flan-T5 ushers in new possibilities for NLP applications. The text-to-text approach of T5, combined with the efficient handling of longer sequences by Flan, enables this hybrid model to deliver superior performance across many NLP tasks. This confluence offers a strong foundation for high-quality language understanding and generation, particularly useful for tasks like document summarization or legal text interpretation where context and sequence length become crucial factors. Flan-T5 also excels in resource-constrained environments. The computational efficiency brought about by Flan's adoption of FAVOR+ means that Flan-T5 can perform at its peak even on devices with limited computing power. This makes it a valuable tool for on-the-go applications, where high processing power is not always available. Despite its many advantages, Flan-T5 isn't without limitations. It requires extensive training to perform at its best, making it resource-intensive. Also, it inherits the limitations of transformer-based models in comprehending some of the subtler nuances of human language, such as sarcasm or implicit context. In conclusion, Flan-T5 marks a significant advance in NLP technology by combining the versatility of T5's text-to-text approach with the efficiency of Flan's FAVOR+ attention mechanism. Although it has certain limitations, its potential applications, especially in resource-constrained environments and tasks requiring long context understanding, are immense. As research progresses, future iterations of Flan-T5 promise even more exciting developments in the realm of NLP.

T5

The T5 (Text-to-Text Transfer Transformer) model, developed by Google Research, is an innovative approach to machine learning models specialized in handling language tasks. Unlike most traditional models which are designed with task-specific architectures, T5 simplifies the process by converting all language processing tasks into a text-to-text format. What sets T5 apart is its unified framework that eliminates the need for task-specific modifications, making it more versatile and efficient than its predecessors. This general-purpose system uses a standard sequence-to-sequence approach where each task, such as translation or summary generation, is executed by "translating" one piece of text into another. At the heart of T5 is the Transformer architecture, an attention-based model that employs self-attention mechanisms for predicting tokens in a sequence considering the full context of input. The Transformer framework allows T5 to handle long-term dependencies effectively, offering improved results on a wide range of tasks compared to previous models. Training T5 involves a two-step process: pre-training and fine tuning. Pre-training leverages unsupervised learning on a large corpus of text data. Here, the model learns to predict tokens in a sentence, thus acquiring a deep understanding of language structure and semantics. Then, during fine-tuning, the pre trained model is adapted to specific tasks using supervised learning on labeled datasets. One of the key innovations of T5 is its use of Causal Language Modeling (CLM), a departure from the traditional Masked Language Model (MLM) used by models like BERT. In CLM, the model is trained to predict the next word in a sentence, given all previous words, which encourages it to develop a more holistic understanding of the sentence context, enhancing its fluency and coherence in generated sentences. T5 also integrates a denoising objective during pre-training, introducing noise into inputs and training the model to reproduce the original, noise-free texts. This process improves the generalization capabilities of the model,making it more robust in handling real-world tasks. The T5 model has demonstrated superior performance on a variety of benchmarks, including the GLUE and SuperGLUE language understanding benchmark suites. Its versatility and task agnostic nature have been a game-changer in natural language processing (NLP), with its impacts felt across machine translation, summarization, question answering, and text classification. However, the T5 isn't without its drawbacks. Its large-scale variants require significant computational resources for training and deployment, making them less accessible for smaller research teams or organizations. Moreover, like most deep learning models, it's often seen as a "black box" due to limited interpretability of its internal decision processes. Overall, the T5 model represents a major stride in NLP. It illustrates the power of transfer learning and the potential of unifying different NLP tasks under one model architecture. As researchers continue to refine and build upon this technology, we can expect to see even more sophisticated and capable language models in the future.

LLaMA

The LLaMA (Lifelong Learning Machine) model represents a significant evolution in artificial intelligence. It embodies a key shift from traditional AI models that required extensive training and retraining for each new skill or application towards one that is capable of continuous learning, much like humans do. LLaMa leverages a concept known as lifelong learning, aiming to create an AI system that can learn from every interaction, building on past knowledge without forgetting previous lessons. This is also referred to as overcoming 'catastrophic forgetting', a common issue in neural networks where the introduction of new information can lead to the loss of previously learned data. The model is designed to adapt and evolve its understanding over time, allowing it to apply learned skills to new situations and tasks. One of the core principles underpinning LLaMA is the idea of transfer learning. This involves the model applying knowledge gained in one context to another, different context. For instance, language skills learned in English could be applied to another language or mathematical principles could be used to solve physics problems. Such cross-application of knowledge mirrors human cognitive processes and significantly increases the model's efficiency. LLaMA uses meta-learning algorithms which guide the way the model learns based on its experiences. It tries to figure out not just what to learn but how to learn it, adjusting its learning methods in response to new information. Meta-learning is particularly useful when it comes to tasks where the model has minimal prior exposure, enabling it to rapidly adapt its approach based on early experiences with the task. Privacy is another crucial aspect of LLaMA. While it continuously learns from interactions, it is also designed to ensure that this does not compromise user privacy. It accomplishes this through techniques such as differential privacy, which allows the model to learn patterns across many users without identifying specific individuals. Despite its promising potential, the LLaMA model does present certain challenges. One of the most significant is computational resources - the model requires substantial processing power and memory to function effectively. There are also ongoing concerns about its ability to handle complex tasks and situations without human intervention, as well as issues related to bias and fairness. In conclusion, LLaMA represents a major step forward in the development of AI models that can learn and adapt throughout their lifetime, much like humans do. Its approach of continuous learning, transfer learning, meta-learning, and privacy preservation offers promising aspects for both practical applications and further research.

6. Industry research directions

Instruction Fine Tuning

In recent years, the task of 'instruction following' has become central to AI research endeavors. Instruction finetuning is a promising area of investigation that seeks to augment the language model's ability to understand and execute complex task descriptions accurately. This entails training an AI model not only to comprehend instructions but also to implement them correctly. The goal of current industry research is to create AI systems that can handle a broader range of tasks, adapt to new situations more efficiently, and demonstrate improved generalization skills by effectively interpreting and executing novel instructions. One direction of research within this niche is focused on data collection methodologies. It involves the development of vast and diverse datasets comprising varied multi-step instructions. The end goal is an AI system capable of understanding and applying directives across different domains, hence enhancing its usability. An example could be designing conversational agents that follow cooking recipes, assembly instructions, or guide users through software troubleshooting steps. Another significant aspect of instruction finetuning studies involves optimizing training procedures. Researchers are exploring ways to create more sophisticated training regimes that leverage reinforcement learning, unsupervised learning, and supervised fine-tuning. For instance, they may use reinforcement learning to reward the model when it successfully follows instructions, thereby refining its future performances. Additional focus areas include improving the granularity of finetuning and developing advanced evaluation metrics. Granularity improvement involves fine-tuning models at the level of individual neurons or smaller subnetworks, which could potentially enhance the model's capacity to learn specific tasks without disrupting its overall performance. On the other hand, creating specialized metrics will ensure researchers can accurately measure a model's ability to interpret and carry out instructions, thus providing clear benchmarks for comparison and progress tracking. A promising yet underexplored avenue is the incorporation of domain-specific knowledge during instruction finetuning. While most current techniques rely on generic pre-training data, customizing pretraining with domain oriented information might yield models that outperform in specialized tasks. Moreover, instruction finetuning can also be combined with other AI techniques such as meta-learning to create more adaptable systems that can quickly learn from new instructions. Meta-instruction learning, where the model is taught how to learn from instructions, could be another ground-breaking research path. Lastly, making AI models robust and interpretable is a crucial research direction. It involves creating methods to ensure that models don't misinterpret or overgeneralize instructions, and are capable of providing clear explanations about their decisions and actions. Overall, instruction finetuning presents an exciting frontier for industry research. It aligns with the broader goal of creating AI systems that mirror human-like comprehension and execution capabilities. The current progress in this area hints at its potential to shape the future evolution of AI, opening up possibilities for applications in various sectors, including customer service, software development, healthcare, and many others. As we continue to explore these prospects, it's clear that the journey will bring both challenges and opportunities, but the promise of advancements in Artificial Intelligence makes it undeniably worth the effort.

Knowledge Grounded

The term "knowledge-grounded" pertains to a system that leverages vast pools of information during its operations. In the context of Artificial Intelligence (AI) and Machine Learning (ML), this refers to models that leverage external knowledge bases or their internal representations of the world to generate responses or make decisions. There is an exciting shift in industry research towards knowledge-grounded approaches, and it's essential to understand its trajectory. As we move further into the 21st century, data plays a pivotal role in shaping our world. The internet has evolved into a colossal repository of information about almost every conceivable subject. Yet, harnessing this wealth of knowledge effectively remains a challenge. That's where knowledge-grounded AI comes into play. These systems can understand, learn from, and use this vast trove of digital knowledge to enhance their performance, making them more effective and efficient. This shift towards knowledge-grounded systems in industry research is driven by several key factors. Firstly, there's a growing demand for more sophisticated AI applications that can handle nuanced tasks, such as customer service, content moderation, or personalized recommendations. Traditional machine learning models often fall short in these areas because they lack the necessary understanding of worldly context. Knowledge-grounded models can fill these gaps by utilizing information from external sources and applying it to the task at hand. Secondly, technological advancements have made it easier to store and process larger amounts of data. This means companies can feasibly build knowledge-grounded models that draw upon extensive online databases and encyclopedias. These resources offer a rich source of information that can significantly improve an AI model's performance. Moreover, advances in natural language processing (NLP) technology, neural network architecture, and transfer learning have enabled machines to understand textual data in more profound ways. They can now comprehend abstract concepts, identify connections between different pieces of information, and use this understanding to generate more accurate, meaningful responses. Yet, with all its potential, the move towards knowledge-grounded systems is not without challenges. For instance, ensuring that

AI models can accurately interpret and apply external information in a relevant context remains a significant hurdle. There's also the issue of data privacy and security - given that these systems often rely on large-scale public databases, ensuring the safe and ethical use of this data is paramount. In conclusion, the industry research direction is progressively inclining towards knowledge-grounded AI. With an aim to develop smarter machines that can tap into the internet's vast knowledge reserves, organizations are investing heavily in this niche field. Although challenges exist, the potential benefits of knowledge-grounded models in enhancing AI capabilities are immense. As we forge ahead, breakthroughs in this domain will undoubtedly shape the future of AI and machine learning.

Reinforcement Learning with Human Feedback

Reinforcement Learning (RL) has emerged as a potent approach to develop self-learning systems that can improve their decision-making capabilities over time. By incorporating human feedback, we can harness the power of RL while reducing negative side effects and unintentional behavior due to reward mis-specification and lack of real-world grounding. A significant research direction in this domain focuses on enabling RL algorithms to comprehend and incorporate human feedback more effectively. This includes developing techniques for active query of humans, where an algorithm asks questions during training to increase its understanding and performance. Additionally, methods are being developed to allow the system to understand and apply feedback from multiple sources which may have discrepancies or conflicts. Another key area of interest is the development of safe exploration strategies. RL agents often learn through trial and-error, but in many real-world scenarios, errors can lead to harmful consequences. Innovations in off-policy learning, safe exploration, and conservative policy updates aim to mitigate risks without hampering the learning process. The incorporation of inverse reinforcement learning (IRL) into the RL with human feedback framework is also promising. IRL enables the AI to infer the underlying values or goals driving human behavior, providing another layer of context. The challenge lies in evolving models that can handle the complexity and subtlety of human behavior and adapt these insights in diverse contexts. To effectively leverage human feedback, it's crucial to reduce the burden on the human trainer. Methods such as batch active learning, where queries are grouped together, and the use of synthetic or simulated feedback generated by other AI models, are emerging as solutions to this issue. Further research is needed in dealing with biases that can emerge from human feedback. Humans are not perfect and their feedback can reflect personal biases or mistakes. Developing mechanisms to identify and correct for such biases is essential for creating fair and balanced RL systems. Reward modeling represents another promising research avenue. Instead of manually specifying a reward function, the idea is to learn it from human feedback. This could involve approaches where the agent generates its own hypotheses about what actions are correct and checks these against human feedback. The integration of RL with human feedback has far-reaching implications for industries such as robotics, healthcare, education, and gaming, where personalized, adaptive decision-making is invaluable. Yet, significant research challenges remain, including developing more efficient algorithms, handling sparse and noisy feedback, and ensuring that systems generalize well in unseen scenarios. In conclusion, Reinforcement Learning with Human Feedback presents an exciting direction for industry research, capable of bridging the gap between algorithmic learning and human intelligence. By combining the problem-solving capabilities of RL with the nuanced understanding provided by human feedback, we can create AI systems that are both powerful and reliable. The key lies in developing robust methods for interpreting and integrating human input while maintaining system safety, fairness, and efficiency.

Symbolic AI Reasoning

Symbolic Artificial Intelligence (AI) reasoning, which focuses on representing problems and their solutions as symbols, is engaging significant momentum in the industry. It has the potential to revolutionize various sectors including healthcare, finance, security, and more. One key direction is the creation of hybrid AI models where symbolic reasoning is integrated with other AI paradigms, especially machine learning techniques. While symbolic AI excels at tasks requiring logical reasoning and explicit knowledge representation, it often falls short when dealing with noisy, unstructured data. Conversely, machine learning, a subset of AI that learns from data, is highly effective with such data but struggles to provide explainability. By combining these approaches, researchers aim to create an AI system that leverages the strengths of both. Another promising direction is enhancing explainability and interpretability in AI systems. As AI permeates more aspects of our daily operations, understanding why AI systems make certain decisions becomes crucial. Symbolic AI, with its transparent rules and representations, provides a way forward. The focus is to develop AI systems that not only make accurate predictions or decisions but also explain their thought process in a manner understandable to humans. Scalability is another area of focus in symbolic AI reasoning. Current symbolic systems struggle to scale up and handle complex, real-world problems. Therefore, efforts are concentrated on developing more efficient algorithms and architectures for symbol manipulation, as well as effective methods of pruning unnecessary search space. The application of symbolic AI in natural language processing (NLP) is another major research direction. Recent advancements in NLP, largely driven by machine learning approaches, have significant limitations in terms of understanding the deeper, often implicit, meanings in human language. Symbolic AI can model the underlying semantics and context behind words and phrases, offering the potential for a more nuanced understanding of language. The increasing integration of AI into critical infrastructure systems, such as power grids, air traffic control, and healthcare systems, brings forth the necessity for robustness and reliability in AI systems. Symbolic AI reasoning, with its inherent ability to follow predefined rules strictly, can provide deterministic outcomes essential in these high-stakes settings. In addition, research is also focusing on how to make symbolic AI systems more adaptable and capable of learning new concepts and rules autonomously. This involves developing mechanisms for symbolic AI to generate new symbols and rules from observational data or through interaction with its environment, thus evolving over time without human intervention. Lastly, there's rising interest in using symbolic AI reasoning for ethical decision making. As AI takes on roles involving significant ethical implications, like autonomous vehicles and judicial decision support systems, it becomes crucial to represent and reason about ethical and moral considerations explicitly. Researchers are exploring ways to encode these principles into symbolic AI systems. Industry research is driving advancements in symbolic AI reasoning, opening up new possibilities and applications. These directions elucidate a future where symbolic AI plays a fundamental role, enabling the creation of AI systems that are not only intelligent but also transparent, reliable, and ethical.

III. PROJECT MANAGEMENT PLAN

1. Overview

Over the past 14 weeks, our team has embarked on an ambitious journey to develop a symbolic dialogue system for general domain state tracking. Our approach has been guided by a deep understanding of the intricacies of symbolic dialogue modeling, coupled with a commitment to leveraging cutting-edge techniques and methodologies.

Initial Phase: Laying the Foundation (Weeks 1-3)

Our initial efforts focused on establishing a solid foundation for our project. We immersed ourselves in the vast body of literature related to symbolic dialogue and general domain state tracking, meticulously examining various methodological approaches and gaining a comprehensive understanding of the challenges and opportunities that lay ahead.

Data Acquisition and Preprocessing (Weeks 2-4)

Recognizing the importance of high-quality data, we dedicated significant time and effort to acquiring and preprocessing the Schema-Guided Dialogue Dataset from Google. This involved meticulously cleaning and normalizing the data, ensuring its compatibility with the Flan-T5 architecture, and structuring it to facilitate effective training.

Model Selection and Optimization (Weeks 3-7)
With a clear understanding of the task at hand and the available data, we embarked on the crucial process of selecting and optimizing an appropriate model architecture. After careful consideration, we chose the Flan-T5 architecture, renowned for its ability to handle complex and nuanced natural language interactions. We meticulously fine-tuned the model's parameters, conducting rigorous experiments to assess its performance on various dialogue tasks.

System Integration and Enhancement (Weeks 8-11)
Our focus shifted towards integrating the optimized Flan-T5 model into a comprehensive symbolic dialogue system. We addressed any performance bottlenecks, developed strategies for handling complex dialogue interactions, and implemented mechanisms for maintaining accurate state tracking across multiple turns of a dialogue.

Evaluation and Refinement (Weeks 12-14)
To ensure the effectiveness of our system, we conducted thorough evaluations on a variety of general domain state tracking tasks. We analyzed the system's strengths and weaknesses, identifying areas for further improvement and refining our strategies accordingly.
Dissemination of Knowledge (Weeks 13-14)
We documented our findings and contributions in a comprehensive report, outlining the challenges we faced, the solutions we devised, and the insights we gained throughout the project. Additionally, we prepared a paper summarizing our work for presentation at the [Conference Name] scientific conference in Indonesia.
Throughout our journey, we have demonstrated a strong commitment to collaboration and open communication, fostering a supportive and productive working environment. We have overcome numerous challenges, embraced new ideas, and consistently strived for excellence. Our project has not only produced a valuable contribution to the field of symbolic dialogue modeling but has also enriched our individual knowledge and skill sets. We are proud of the progress we have made and look forward to continuing our exploration of this fascinating and ever-evolving domain.

2. *Work Details*

Our project, "Symbolic Dialogue for General Domain State Tracking," is divided into three distinct phases: the foundational research phase, the model development and training phase, and the ablation study and refinement phase.

Foundational Research Phase (Weeks 1-4)

- **Objective:** Establish a comprehensive understanding of the theoretical underpinnings, related techniques, and relevant data sources for symbolic dialogue modeling and general domain state tracking.

Tasks:

1. Conduct a thorough literature review to gain a deep understanding of symbolic dialogue modeling, general domain state tracking, and related techniques, including natural language processing, machine learning, and artificial intelligence.
2. Identify and evaluate various methodological approaches for developing and training symbolic dialogue systems, considering their strengths, weaknesses, and applicability to the specific task of general domain state tracking.
3. Explore and assess potential data sources, including academic databases, publicly available datasets, simulated dialogue interactions, and legal documents with structured information.
4. Analyze the limitations and biases inherent in the available data sources and develop strategies for mitigating their impact on the model's training and performance.

Team Members:

- All team members will collaborate on literature review and methodological exploration.
- Nguyễn Sơn Tùng will focus on identifying and evaluating data sources, particularly those related to legal domain dialogue.

Model Development and Training Phase (Weeks 5-10)

- **Objective:** Design, develop, and train a symbolic dialogue system capable of effectively tracking state across multiple turns of a dialogue in a general domain setting.

Tasks:

1. Select the most suitable model architecture for the symbolic dialogue system, considering factors such as computational efficiency, accuracy, and adaptability to different dialogue contexts.
2. Perform hyperparameter optimization to fine-tune the selected model's parameters, ensuring optimal performance on the chosen data sources.
3. Implement appropriate training procedures, including data preprocessing, model initialization, and loss function selection, to effectively train the symbolic dialogue system.
4. Conduct a series of experiments on a variety of dialogue tasks to assess the model's performance in tracking state, handling complex interactions, and generating coherent responses.

5. Analyze the experimental results to identify strengths, weaknesses, and areas for improvement in the model's performance.

Team Members:

- Nguyễn Sơn Tùng and Nguyễn Mạnh Tường will collaborate on model selection and hyperparameter optimization.
- Lê Xuân Tùng will focus on implementing training procedures and conducting experiments.

Ablation Study and Refinement Phase (Weeks 11-14)

- **Objective:** Systematically evaluate the impact of individual components of the symbolic dialogue system through ablation study and refine the model architecture and training procedures for enhanced performance.

Tasks:

1. Design and execute a comprehensive ablation study, systematically removing or modifying specific components of the model to assess their contribution to overall performance.
2. Analyze the results of the ablation study to identify the most significant components and understand their individual and synergistic effects on the model's performance.
3. Based on the ablation study findings, refine the model architecture by adding, modifying, or removing components to enhance its overall performance and address identified weaknesses.
4. Adapt and refine the training procedures to optimize the model's training process and ensure effective learning of the refined architecture.
5. Evaluate the performance of the refined model through experiments and compare it to the original model to assess the effectiveness of the ablation study-driven refinements.

Team Members:

- Nguyễn Sơn Tùng will focus on designing and executing the ablation study.
- Nguyễn Mạnh Tường will focus on analyzing and interpreting the ablation study results.
- Lê Xuân Tùng will focus on refining the model architecture and training procedures based on the ablation study findings.

Task Distribution Table:

Week	Team Member	Task

1-4	All team members	Conduct a comprehensive literature review, explore various methodologies, and identify potential data sources suitable for the project.
4	Nguyễn Sơn Tùng	Thoroughly evaluate the identified data sources, assessing their quality, relevance, and potential biases.
5-6	Nguyễn Sơn Tùng and Nguyễn Mạnh Tường	Collaboratively select appropriate models for the project, considering factors such as model complexity, performance, and compatibility with the chosen datasets. Optimize hyperparameters for the selected models to enhance their performance.
7-8	Lê Xuân Tùng	Implement the training procedures for the selected models, ensuring proper data preparation and model configuration. Conduct experiments to evaluate the performance of the trained models under various conditions.
9-10	All team members	Analyze the experimental results obtained from the trained models, evaluating their performance metrics, identifying patterns, and drawing meaningful conclusions.
11	Nguyễn Sơn Tùng	Design an ablation study to identify the key components that contribute most significantly to the performance of the selected models.
12	All team members	Execute the ablation study by systematically removing or modifying specific components of the models, observing the impact on their performance.
13	Nguyễn Mạnh Tường	Analyze the results of the ablation study, assessing the contribution of each component to the overall performance of the models.
14	Lê Xuân Tùng	Based on the findings of the ablation study, refine the model architecture and hyperparameters to improve performance. Evaluate the refined models to assess the effectiveness of the improvements.

Team management software

Effective project execution hinges on a well-coordinated team that can seamlessly collaborate, manage tasks, and track progress efficiently. To achieve this, our team employed a carefully selected arsenal of team management software, each playing a distinct role in streamlining our workflow and ensuring project success.

Notion: A Unified Knowledge Hub for Project Information

Notion, a versatile workspace management tool, served as our central knowledge base, providing a comprehensive repository for project documentation, research findings, and team discussions. Its intuitive interface and robust features allowed us to organize, manage, and access a vast amount of information with ease, fostering a shared understanding among team members.

- Project Documentation: We maintained a comprehensive project wiki in Notion, encompassing a detailed project overview, objectives, methodology, timelines, and progress tracking. This centralized repository ensured that everyone had easy access to the latest project information, minimizing the need for repetitive explanations and facilitating informed decision-making.
- Literature Reviews and Shared Research Findings: Notion's collaborative features facilitated the organization and sharing of literature reviews and research findings. Each team member could contribute to and access the reviews, ensuring that everyone was on the same page regarding the relevant research. This collaborative approach streamlined the research process and enabled us to leverage each other's expertise effectively.
- Meeting Notes and Action Items: Notion's note-taking capabilities proved invaluable for capturing meeting notes, action items, and follow-ups. By centralizing this information in Notion, we ensured that key discussions and decisions were easily accessible to all team members, promoting accountability and ensuring that tasks were completed promptly.

Overleaf: Real-time Collaboration for Cohesive Paper Writing

Overleaf, a cloud-based collaborative writing platform, emerged as an essential tool for coordinating paper writing tasks. Its real-time collaboration features and intuitive interface allowed multiple team members to work on the same document simultaneously, streamlining the writing process and ensuring consistency in style and formatting.

- Real-time Collaboration: Overleaf enabled seamless collaboration, allowing team members to edit and comment on the paper simultaneously. This real-time feedback loop facilitated a more efficient writing process, enabling us to quickly incorporate suggestions and maintain a cohesive writing style.
- Version Control: Overleaf's robust version control system kept track of all changes made to the document, ensuring that we could easily revert to previous versions if necessary. This feature proved particularly useful when addressing conflicting edits or incorporating feedback from external reviewers.

- Citation Management: Overleaf's built-in citation management tool simplified the process of adding and formatting citations. Its integration with various citation styles and its ability to automatically generate bibliographies saved us valuable time and effort, ensuring that our paper adhered to strict academic standards.

ClickUp: Task Management and Project Progress Tracking

ClickUp, a versatile project management tool, played a pivotal role in task management and project progress tracking. Its intuitive interface and robust features enabled us to effectively organize tasks, monitor progress, and identify potential roadblocks.

- Task Management: ClickUp's task management capabilities allowed us to create, assign, and track tasks with clear deadlines and dependencies. This granular task management ensured that everyone was aware of their responsibilities and the overall project timeline, preventing delays and fostering accountability.
- Progress Tracking: ClickUp's Kanban boards and Gantt charts provided clear visualizations of project progress, enabling us to identify any potential bottlenecks or deviations from the planned schedule. This visual representation of progress facilitated proactive decision-making and allowed us to take corrective actions when necessary.
- File Sharing: ClickUp's file-sharing functionality eliminated the need for external file-sharing platforms, streamlining access to project-related documents. By storing all relevant files within ClickUp, we ensured that everyone had easy access to the latest versions of documents, reducing confusion and delays.

The combination of Notion, Overleaf, and ClickUp proved to be an effective team management strategy for our project. These tools enabled us to organize our knowledge, collaborate efficiently, track progress, and ultimately deliver a successful project. By leveraging the strengths of each tool, we fostered a collaborative and productive work environment, enabling us to overcome challenges and achieve our project goals.

IV. MATERIALS AND METHODS

1. Materials

In the scope of this capstone, we choose Schema-Guided Dialogue (SGD) as our main exploration for the experiments. The SGD dataset is a large-scale, multi-domain, task-oriented dialogue dataset that consists of over 20,000 annotated conversations between a human and a virtual assistant. These conversations involve interactions with services and APIs spanning 20 domains, such as banks, events, media, calendar, travel, and weather. For most of these domains, the SGD dataset contains multiple different APIs, many of which have overlapping functionalities but different interfaces, which reflects common real-world scenarios. We adopted SGD instead of previous datasets (like MultiWoz) for several reasons:

Largest multi-domain schema-based dataset: The multi-domain nature of the SGD dataset plays a crucial role in enabling the system to generalize to new domains and tasks. By exposing the dialogue system to conversations from a wide range of domains, the system can learn to identify patterns and transferable knowledge that apply to broader conversational contexts. This ability to generalize is essential for real-world applications, where users may interact with the system in various domains and contexts. Moreover, the SGD dataset's multi-domain nature facilitates the development of domain-agnostic dialogue systems. By training the system on conversations from multiple domains, it can learn to extract common patterns and conversational strategies that apply across different domains. This domain-agnostic approach reduces the need for domain-specific training data, making it more efficient and cost-effective to deploy dialogue systems in new domains. In summary, the SGD dataset's large size and multi-domain nature provide a valuable foundation for training and evaluating robust and generalizable task-oriented dialogue systems. By exposing the system to a diverse range of conversational contexts, the SGD dataset enables the system to handle a variety of tasks and interactions, effectively adapt to new domains, and generalize to broader conversational scenarios.

Schema-guided: The SGD dataset includes schemas that define the structure of the dialogues, providing a clear and organized representation of the conversation flow. These schemas outline the expected intents, slots, and relations within each domain, enabling the dialogue system to better understand the context of the conversation and track the progression of user goals. This structured representation simplifies the task of state tracking, ensuring that the system maintains a consistent understanding of the conversation's purpose and the user's intentions. The schema-guided nature of the SGD dataset makes it easier to understand and explain the system's decisions. By aligning the system's actions and responses with the predefined schema, the decision-making process becomes more transparent and interpretable. This explainability is crucial for building trust with users and enabling debugging and error analysis. The SGD dataset is richly annotated with slot values, intent labels, and dialogue state information. These annotations facilitate supervised learning, providing the training data necessary for the dialogue system to learn the relationships between user utterances, intents, and slots. The annotations also enable comprehensive evaluation, allowing for detailed assessment of the system's performance in various aspects of dialogue processing, including intent recognition, slot filling, and state tracking.

Realistic: Exposure to realistic dialogues allows the system to learn the nuances of human language and the natural flow of conversations. By observing how users interact with virtual assistants in real-world settings, the system can learn to generate responses that sound natural, fluent, and contextually appropriate. This ability to produce natural and engaging responses is crucial for fostering positive user experiences and building trust in the system. Realistic dialogues capture the complexities and challenges of real-world conversations, such as ambiguous utterances, incomplete information, and unexpected user requests. By training on these realistic dialogues, the system develops the ability to handle these challenges effectively. It can learn to identify and resolve ambiguities, infer missing information, and adapt to unforeseen conversational situations. The SGD dataset's realistic dialogues encompass a wide range of user interactions, including a variety of conversational styles, intentions, and goals. This exposure to diverse user interactions prepares the system to handle the diversity of real-world users and their communication patterns. It can learn to adapt its responses and behavior to different personalities, communication styles, and interaction contexts. Realistic dialogues provide a valuable resource for error detection and analysis. By observing how the system performs in real-world scenarios, developers can identify areas where the system struggles and make targeted improvements. This feedback loop is essential for continuous improvement and ensuring that the system is well-equipped to handle the complexities of real-world interactions.

Because of being well-annotated, SGD dataset can be used for a variety of tasks, including:

- Dialogue State Tracking: Tracking the state of the conversation and maintaining a consistent representation of the user's goals and intentions.
- Intent Recognition: Identifying the user's intent from their utterances.
- Slot Filling: Extracting slot values from the user's utterances.
- Response Generation: Generating appropriate and informative responses to the user's utterances.
- Policy Learning: Learning policies that guide the dialogue system's actions to achieve the user's goals.

Overall, the Schema-Guided Dialogue dataset is a valuable resource for developing task-oriented dialogue systems that are accurate, versatile, and user-friendly.

Research on TOD has predominantly been focusing on developing a single module or end-to-end LLM. A more unified system is necessary to enhance overall performance, robustness, and efficiency. LLM is the heart of all the advancements. But, LLMs are trained with natural language, exposing a massive downside, ambiguity. Natural language is common in daily conversation. Natural language is flexible with a vast amount of additional attributes: emotion, context, ... In case of insufficient context, natural language is ambiguous. Humans can add context by clarifying terms and expressions. Machines, on the other hand, are not able to provide context themselves, but depend on external sources. Programming language is quite the opposite compared to Natural language. Programming language respects a set of robust syntax. Thus, there is no risk of mismatch or confusion using a programming language. A simple solution for TOD is to combine the advantages of these 2 language types, specifically, the flexibility of Natural language and the disambiguation of Programming language. We believe a domain-specific language (DSL) for TOD can open a new era for the TOD system. Behind every human utterance are actions to reach a target with additional clarification. Breakdown attributes and behavior significantly simplify downstream task automation.

2. Methods

a. Schema-based Retrieval TOD

These are indeed essential criteria for a task-oriented dialogue system to effectively assist users in completing their tasks. Let's delve into each requirement in more detail:

Requirement 1: Schema-based

A schema-based approach provides a well-organized and methodical framework for the dialogue system to comprehend and effectively address user requests. This structured and systematic approach ensures that the system possesses the capability to handle an extensive array of diverse tasks spanning across multiple domains, as long as these tasks are defined within the specified schema. As elucidated in the comprehensive literature review, schema-based approaches offer a multitude of advantages, including but not limited to: (1) Cross-domain applicability, wherein the system can be seamlessly adapted to cater to various domains by meticulously defining the appropriate schemas for each specific domain. (2) Facilitation of structured interaction, whereby the system adeptly guides the user through the given task by furnishing well-crafted prompts and a wide range of options that are meticulously crafted based on the defined schema. (3) Mitigation of errors and reduction of misunderstandings, as the inherent structured nature of the schema ensures that the conversation remains focused on the task at hand, thereby significantly minimizing the occurrence of errors and misunderstandings that may arise during the interaction process.

Requirement 2: Out-of-scope aware

Real-world conversations can often be complex, characterized by frequent context switching and the inclusion of irrelevant information. To ensure a seamless and productive dialogue experience, a dialogue system must possess the ability to adeptly navigate and manage these situations. Firstly, the system should employ robust context switching detection mechanisms, enabling it to recognize instances when the user deviates from the current task and assess the relevance of the diversion. This entails the system's capacity to discern whether the user's shift in context is pertinent to the ongoing conversation or not. Secondly, in cases where the context proves to be irrelevant, the system should exhibit a polite and considerate approach, acknowledging the diversion without causing any disruption to the flow of the conversation. It should gracefully guide the user back to the primary task at hand, ensuring that the interaction remains focused and productive. Finally, when the context is deemed relevant, the system should possess the capability to seamlessly incorporate it into the ongoing task, if applicable. This involves leveraging the pertinent context to enhance the task execution or, if necessary, providing additional information to further enrich the user's understanding. By effectively addressing context switching and handling irrelevant information, the dialogue system can maintain a natural and engaging conversation, fostering a more efficient and satisfying user experience.

Requirement 3: Extendable

A modular and extensible design serves as a cornerstone for the evolution and advancement of individual components within the dialogue system, while ensuring the overall system remains unaffected. This inherent flexibility is paramount for fostering continuous improvement and adaptability in response to evolving requirements and technological advancements. Firstly, the modular design empowers developers to create and enhance specific modules independently, enabling them to be developed, tested, and updated in isolation from other components. This not only expedites the development process but also facilitates easier maintenance, as modifications or optimizations can be focused on specific modules without necessitating extensive changes to the entire system. Such modularity enhances the system's scalability, making it easier to incorporate new features or address specific functionality requirements without compromising the stability or integrity of the system as a whole. Moreover, the extensible nature of the design allows for seamless substitution of modules with improved or alternative versions. This means that as novel technologies or more efficient algorithms emerge, individual modules can be replaced or upgraded without disrupting the core functionality of the system. This adaptability ensures that the dialogue system can leverage the latest advancements in natural language processing, machine learning, or any other relevant field, without requiring a complete overhaul of the entire system architecture. By embracing component substitution, the dialogue system can continuously evolve and incorporate cutting-edge innovations, ultimately resulting in improved performance, enhanced user experience, and increased system efficiency.

Taking into consideration these criteria, we present an in-depth overview of a minimal Task-Oriented Dialogue (TOD) system consisting of five essential components. The first component, State Tracking (1), plays a crucial role in distinguishing between in-domain information and out-of-scope actions. Particularly in zero-shot settings, where the system is required to handle tasks not explicitly defined in the schema, out-of-scope actions are identified by their absence in the schema. Component (1) serves as the natural language interface for downstream systems and must effectively maximize the information extracted from user utterances. It is designed to extract slot values directly from user inputs, even in the presence of misspellings or abbreviations, ensuring accurate understanding and processing.

In a similar vein, Intent Detection (2) operates in tandem with (1), revealing both the user's intent and the associated slots through the actions expressed in the dialogue. This component utilizes a similar mechanism as (1) to extract relevant information, leveraging the actions specified in the schema. The Dialogue Policy (3) serves as the core decision-making component of the system. It reacts to each action generated by (1) and interacts with the underlying database. It is important to note that (1) does not check for word faultiness, necessitating an extension to address entity linking and ensure accurate information retrieval.

The database, informed by the refined conversation context, is responsible for querying and retrieving relevant information. Each item within the database contains a field that leads to unstructured data documents, enabling comprehensive and contextually-rich responses. The output of the Dialogue Policy (3) consists of abstract actions that guide the response generation process and provide additional item context.

Components (4) and (5), Item-based Response Generation and Document-based Response Generation, respectively, can both be implemented as Language Models (LLMs). However, (4) utilizes prompted item attributes, while (5) relies on knowledge documents [24]. This approach follows the guidelines outlined in REML (Reference to Explicit Memory and Latent Memory) [24]. Additionally, Component (4) utilizes an RLHF (Reinforcement Learning from Human Feedback) approach in conjunction with the LLM model, using prompts provided by the schema to generate responses.

To visualize the system architecture, refer to Figure 3, which provides a detailed visual representation of the individual components and their interconnections.

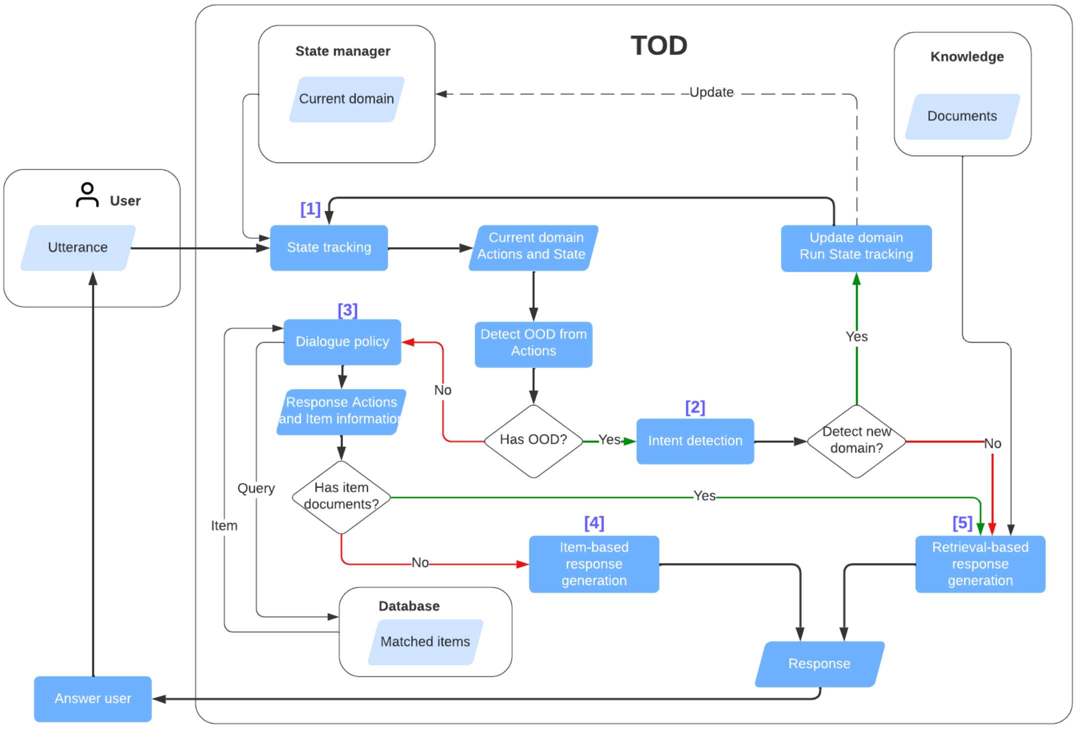

Fig. 3. Overview of TOD system with retrieval capability. (1) capture user utterances to return abstracted actions, slot values, and undefined actions. (2),triggered under undefined actions condition, to examine whether new intent is formed. In case new intent, we restart the flow with the new intent's schema. If the schema is not enough to respond, (5) utilize document-based information to provide a more accurate and knowledgeable response. Supposing state tracking (1) catches no abnormality, dialogue policy (3) uses actions and slot values to interact with external storage. (3) supplies item-based information for (4) or document-based for (5).

b. State tracking In-depth fine-tuning

A symbolized schema in combination with symbolized output forces the model to adapt seamlessly to changes from the schema and understand the DSL coding syntax. The in-depth fine-tuning process refers to the TOD DSL adapting process. By convention, regular fine-tuning refers to training the model with data of the new domain. After the action, the model can perform notably well on fine-tuned tasks. In-depth fine tuning includes data from all relative sub-tasks during the fine-tuning process, thus, the model is allowed to learn the basis of the problem, not just one aspect. As defined in the prior section, state tracking splits a user's natural utterance into two parts: behaviors and attributes. Action are behaviors while slot values are attributes. We design multiple tasks for the model to adapt independently to each part. From observation, behavior spans throughout the dialogue and attribute is temporary in a few dialogue turns. The reactive model understands the user behavior and can decide on response actions without knowing any attribute. For example, after the user is informed of all required slots, the system can decide to query the database without being aware of any slot value. In terms of efficiency, breakdown behavior, and attribute benefits transformer architecture. Actions and states of one turn can easily be detected with affordable input length, and then stored in long-term storage. The decomposition eliminates the need for the complete conversation as input.

Breaking down complex tasks into smaller, more manageable subtasks can significantly enhance the learning efficiency and performance of deep learning models. This decomposition approach offers several advantages. By dividing a complex task into smaller components, the model is presented with less intricate relationships and patterns to learn at once. This simplification reduces the overall cognitive load on the model, making the learning process more manageable and efficient. Additionally, each subtask can be tailored to specific aspects of the overall problem, allowing the model to focus its learning efforts on the most relevant features and relationships. This targeted approach leads to a deeper understanding of each subtask and strengthens the model's overall representation of the problem.

Modular Development: The concept of modular development plays a crucial role in the construction of a task-oriented dialogue system, as it allows for the breakdown of the overall task into smaller, more manageable modules. By decomposing the task into distinct components, developers can focus on the independent development and testing of each module, promoting flexibility and facilitating advancements in the field. This modular approach empowers researchers to explore and experiment with various architectures and training techniques for each subtask, leading to the discovery of more effective and optimized solutions. The modular development methodology enables researchers to delve into the intricacies of each component, honing in on specific challenges and exploring novel approaches to address them. For instance, researchers may experiment with different neural network architectures, such as recurrent neural networks (RNNs), transformer models, or graph neural networks, to determine the most suitable architecture for a particular module. Moreover, researchers can apply diverse training techniques, including supervised learning, reinforcement learning, or even transfer learning, to improve the performance and robustness of individual modules. The modularity aspect provides researchers with the freedom to iterate and refine each module independently, without impacting the overall system's functionality. This agility allows for rapid prototyping, as researchers can experiment with various ideas and iterate on different components, leading to incremental improvements over time. It also encourages collaboration within the research community, as researchers can share and compare their module implementations, fostering an environment of knowledge exchange and collective advancement. Additionally, the modularity of the development process promotes code reusability, as researchers can encapsulate the logic and functionality of each module into separate software components. This reusability not only enhances the efficiency of development but also enables easier maintenance and future extensions.

Researchers can leverage existing modules or adapt them for other dialogue systems, reducing redundancy and accelerating the development cycle.

Improved Generalization: Through the process of learning to solve smaller, well-defined subtasks, the model acquires a more profound understanding of the underlying principles and patterns that govern the problem domain. This focused learning approach equips the model with a generalized knowledge that extends beyond specific instances, enabling it to adapt to new situations and effectively generalize its understanding to previously unseen data. By breaking down the problem into smaller subtasks, the model can concentrate on mastering the intricacies of each component, gradually building a comprehensive understanding of the problem domain. This targeted learning allows the model to discern the fundamental patterns, relationships, and dependencies that exist within the data, thereby establishing a solid foundation of knowledge. As the model gains proficiency in solving these well-defined subtasks, it acquires a deeper intuition for the problem at hand. This generalized knowledge acquired through subtask learning provides the model with a robust framework for handling new situations and unseen data. It equips the model with a repertoire of learned concepts, enabling it to draw upon its understanding of the underlying principles to make informed predictions and decisions in novel scenarios. By leveraging the patterns and principles it has learned, the model can extrapolate from its existing knowledge to generate meaningful responses or take appropriate actions in previously unencountered contexts. Moreover, the process of learning smaller subtasks fosters a more interpretable and explainable model. By focusing on specific problem facets, the model's internal representations can be analyzed and understood more effectively. This interpretability allows researchers and developers to gain insights into how the model is learning and reasoning, enabling them to provide explanations or make improvements based on this understanding. Furthermore, the ability to generalize knowledge across subtasks enhances the model's adaptability to changes or variations within the problem domain. As new data or scenarios emerge, the model can leverage its foundational understanding to quickly adapt and make accurate predictions or decisions. This adaptability is particularly valuable in dynamic or evolving environments where the model needs to continuously learn and adapt to new information.

Error Localization and Correction: The decomposition of a task into smaller subtasks offers a significant advantage when it comes to error identification and debugging within a complex system. By breaking down the task into its constituent components, errors can be more easily localized and traced back to specific subtasks. This localization capability enables developers to engage in targeted debugging and make improvements to individual components, ultimately leading to the development of more robust and accurate solutions. When errors occur in a complex system, the ability to pinpoint their origin is paramount for efficient troubleshooting. By decomposing the task into modular subtasks, each with its unique set of responsibilities, the system's behavior and output can be analyzed at a granular level.

This fine-grained examination allows developers to isolate the subtask or component that is responsible for the error, reducing the overall complexity of the debugging process. The localization of errors to specific subtasks simplifies the debugging workflow by providing a clear starting point for investigation. Developers can focus their efforts on the identified subtask, thoroughly examining its input, output, and internal processes. This targeted approach allows for a more efficient allocation of resources, as debugging efforts can be concentrated on the specific component where the error originated. Furthermore, the localization of errors facilitates a deeper understanding of the system's behavior and performance. By closely examining the interactions and outputs of individual subtasks, developers can gain insights into the specific challenges and limitations faced by each component. This understanding helps in identifying common error patterns, uncovering potential bottlenecks, and refining the design and implementation of the subtasks to minimize future errors. The targeted debugging and improvement of individual components not only enhance the overall accuracy of the system but also contribute to its overall robustness. By addressing errors at the subtask level, developers can iteratively refine and optimize each component, ensuring that they perform their intended functions reliably. This iterative process of debugging and improvement leads to a more stable and resilient system, capable of handling a wide range of scenarios and inputs without compromising its performance.

Scalability: The process of breaking down a task into smaller modules not only facilitates error identification and debugging but also offers significant advantages when it comes to scaling the model to handle larger and more complex problems. By decomposing the task into modular subtasks, the model gains flexibility and adaptability, enabling it to address new challenges without the need for a complete overhaul of the architecture. As the complexity and scale of a problem increase, it becomes essential to ensure that the model can effectively handle the additional demands. By breaking down the task into smaller modules, each with its specific functionality, the overall system can be designed in a way that allows for easier scalability. New subtasks can be added or existing ones can be modified without disrupting the entire architecture, providing a more efficient and sustainable approach to tackling larger and more complex problems. The modular nature of the system allows for seamless integration of new subtasks to address emerging challenges. When faced with new requirements or novel problem domains, developers can introduce additional subtasks that cater to these specific needs. This modular expansion minimizes the impact on the existing architecture, as the new subtask can be integrated into the system without requiring substantial modifications to the already established components. This adaptability enables the model to evolve and grow alongside the problem domain, ensuring its continued relevance and effectiveness. Furthermore, the ability to modify individual subtasks within the modular framework provides a powerful mechanism for fine-tuning the model's capabilities. As new insights are gained or improvements are identified, developers can refine or replace specific subtasks to enhance the model's performance. This targeted modification allows for incremental improvements without disrupting the overall system, enabling a more agile and iterative development process. The scalability afforded by the modular design also extends to computational resources. By breaking down the task into smaller subtasks, the computational requirements can be distributed and optimized, allowing for efficient utilization of available resources. This scalability is particularly valuable when dealing with resource-intensive tasks or when operating in resource-constrained environments. It ensures that the model can handle larger and more complex problems without overwhelming the computational infrastructure.

As in figure 4, we introduce two new tags into the existing tags system from previous papers: target acts and dependencies. Targetacts represent the user's intent, while dependencies represent the actions that must happen to achieve the target acts. We provide additional undefined actions in both user acts and system acts to assist out-of-scope detection. We later refer to models trained with new tags as $IFST_{xtags}$. We believe that being explicit aids the system in deciding the next actions.

Our ultimate goal is to guide the model through an overview idea of TOD. We focus on 3 sub-tasks: see Appendix

• Slot filling. The model focuses on multiple QAs to describe the current state of the dialogue, including available slots with their values. Params and Conversations are given as input.

• Action tracking. Given action-related tags and an utterance, the model must point out what actions are in the utterance. The goal is to guide the model to the foundation of schema actions and practice utterance simplification. The task is applied for both system and user utterances.

• Next action prediction. Traditionally, the Conversation is utilized to recommend the next actions. To plan out for the next move, the system only needs to know abstractly what the conversation is about. Instead of overwhelming the model with a bunch of irrelevant context, we use the dialogue History and action-related tags to propose the next action.

Previously, AnyTOD suggested action as a recommendation to achieve a zero-shot dialogue policy. Still, standard SGD actions are not sufficient to fully represent dialogue policy. We create new actions to the schema, namely, "inform undefined", "request undefined", and "query <domain> api". The query action suggestion is redundant because it can be triggered by Dialogue policy under some condition fulfillment. Yet, the model should know the whole process of TOD. Such redundancy also allows hooks for future work on fully automated Dialogue policy. Likewise, undefined actions are important. The undefined actions are the mechanic to examine the relevance between utterances and the schema. Any action not following the schema is categorized as undefined. Not only operate as the controller for detecting domain switching, but these actions provide tremendous context for learning action nuance. A conversation may consist of multiple domains. Previous domain turns consist of previous domain actions that ought to be clarified for training and inference. During training, we transform all previous domain request actions into requests undefined, similar to inform actions. The number of undefined actions match the map one-to-one with in-domain action. Plus, query actions, depending on domain schema, are removed in case of out-of-scope. These transformations are casual ways for the model to learn out-of-scope context. In the inference phase, the model is expected to explain a few last turns as both state and actions are accumulative and can be stored in memory.

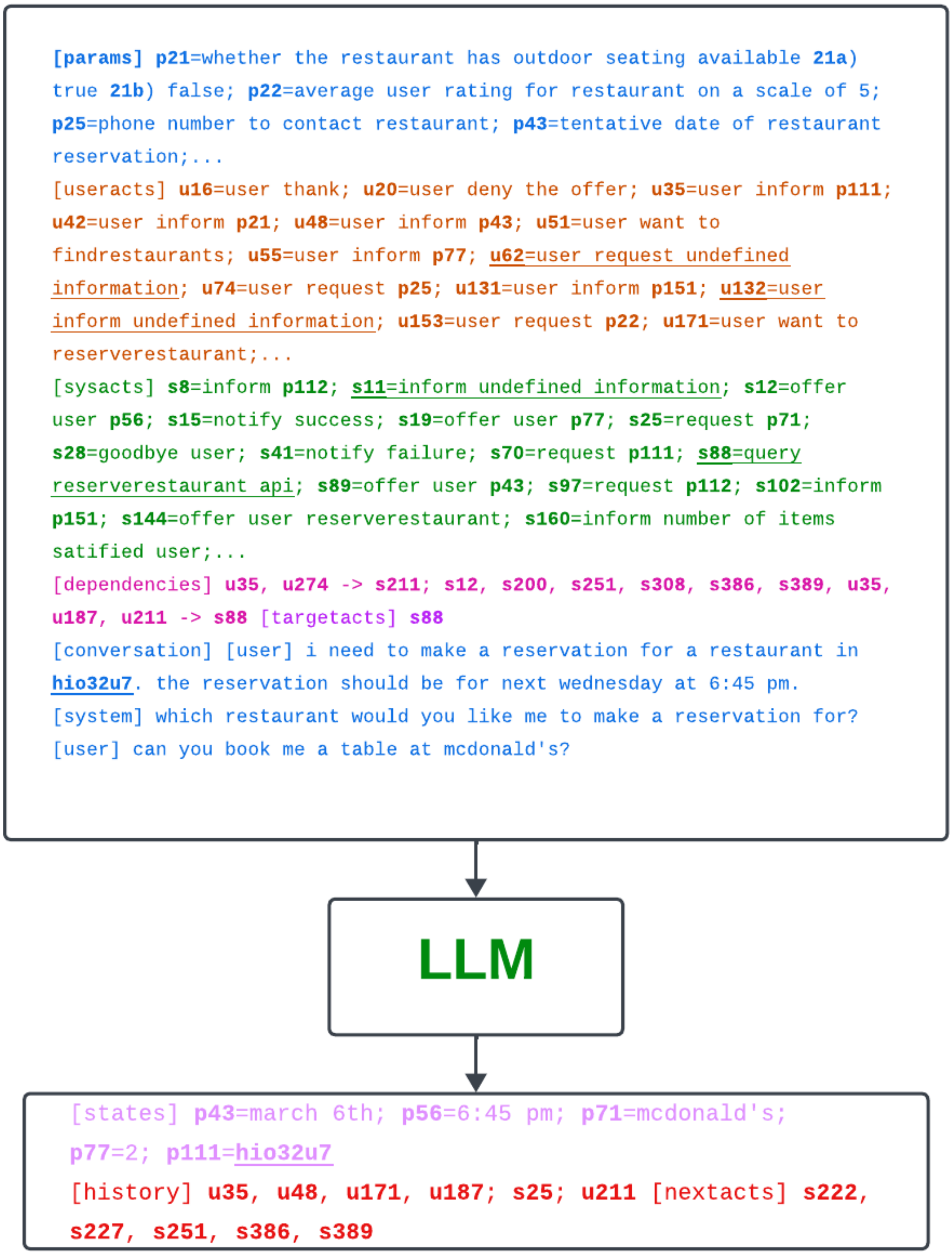

Fig. 4. State tracking example. The underlined components are an example of input and output state tracking.

Moreover, both inbound and outbound information apply the symbolic method. Action names and parameter names symbolization allow training dataset synthesis. With different schema descriptions from SGD-X and symbolized names, the model is capable of learning the exact conversation with multiple domain schemas. Through schema diversification, we aim to gain better generalization. Remark while randomizing ids, we put each tag content in the ascending order. Pix2Seq[25] suggests random output sequence ordering yields the best performance as the regressive model can fix its mistake in later tokens. Yet, we respect single input single output diagrams and leave the discovery for future research. Additionally, language models are known to easily get hallucinations given specific context. The language model for TOD is not an exception. We introduce randomization to slot values across the training dataset, later referred to as *IFSTXrand*. Random slot values are chosen and symbolized for both in output and the input conversation. We believe this would impact the model to learn more about the surrounding context instead of the meaning of the slot value itself.

c. Model

In the realm of task-oriented dialogue systems, encoder-decoder models like T5 hold an advantage over decoder-only architectures when it comes to state tracking. This advantage stems from the inherent design principles and capabilities of encoder-decoder models. Let's delve into the key factors that contribute to their superiority in state tracking tasks. First, T5-like models are more flexible in terms of input and output format.Encoder-decoder models, including T5, are trained on massive datasets of structured text and code. This exposure to structured input and output sequences equips these models with the ability to effectively handle the stateful nature of task-oriented dialogues. They can efficiently track the state of the conversation, maintaining consistency across multiple turns. Secondly, T5 utilizes cross-attention with separate encoding and decoding, lessening lengthy context loss. The encoder-decoder architecture clearly separates the processes of understanding the input context and generating the output response. This separation allows for the implementation of state tracking mechanisms within the decoder component. The decoder can explicitly maintain and update state information based on the conversation history, enabling effective state tracking. Encoder-decoder models like T5 incorporate sophisticated attention mechanisms that empower the model to focus on both the current input and the preceding conversation history. This ability to selectively attend to relevant information is crucial for state tracking, as it allows the model to maintain a coherent understanding of the task and the user's intentions. Lastly, the T5 family, especially flan-T5, has distinguished fine-tuning capabilities for Domain Adaptation. T5 possesses the ability to be fine-tuned on specific task-oriented dialogue datasets. This fine-tuning process further enhances its state tracking capabilities by adapting the model to the specific domain and task requirements. As a result, T5 can track state more effectively in a wider range of task-oriented dialogue scenarios.

In essence, the combination of structured input and output processing, clear separation of encoding and decoding, sophisticated attention mechanisms, and fine-tuning capabilities makes encoder-decoder models like T5 better suited for state tracking in task-oriented dialogue systems compared to decoder-only architectures. These factors enable T5 to effectively maintain contextual awareness, track the conversational flow, and generate consistent and relevant responses throughout the dialogue. Therefore, we believe T5 is the best architecture to serve state tracking.

d. Metrics

Out-of-scope, in schema-guided TOD, is the situation when the schema is no longer enough to handle the situation (e.g. domain switching and open domain). SGD consists of multi domain dialogues. When an action is not represented in the schema but exists in user utterance, the system detects a domain switch. Those actions are classified as user-undefined actions. User undefined actions F1 (UUAF1) are the key metrics to influence out-of-scope effectiveness. After investigation, UUAF1 works best when tracked at each turn instead of each undefined action within a turn. Each turn outcome is:

$$outcome = \begin{cases} Positive & \text{for undefined\_action in turn} \\ Negative & \text{for other case} \end{cases} \qquad (1)$$

outcome = (*Positive* for undefined action in turn *Negative* for other case (1) For each dialogue, F1 is calculated using outcome of each turn 1. Recall F1 formula is

*F*1 = Precision × Recall Precision + Recall

. Then, we work out UUAF1 as the average of all dialogue F1. We track undefined action availability within a turn and calculate the F1 score across the dialogue. In the scope of this research, evaluation metrics for model performance are joint goal accuracy for slot filling, action F1 (AF1) for action tracking, system action F1 (SaF1) for next action prediction, and UUAF1 for out-of-scope detection. These metrics are inspired by AnyTOD, yet, there are technical differences. First, the action F1 score includes both user action tracking and system user tracking. Second, SaF1 records system queries and informs actions, instead of capturing all SGD actions. We track only essential actions to focus on modular performance, other actions are the responsibility of Dialogue Policy. For SaF1 and AF1, outcome for each turn is:

$$outcome = \begin{cases} Positive & \text{for predict\_actions match label\_actions} \\ Negative & \text{for other case} \end{cases} \tag{2}$$

outcome = (*Positive* for predict actions match label actions *Negative* for other case (2)

Joint Goal Accuracy (JGA) is the common metric for slot filling calculation, and also the hardest metric which utilizes Fuzzy string matching. Fuzzy string matching, also known as approximate string matching, is a technique for finding strings that match a given pattern approximately rather than exactly. This is useful for tasks such as spell checking, autocompletion, and natural language processing, where it is often important to find matches even when the input strings contain typographical errors or other minor variations. Common fuzzy string method is Levenshtein distance. This algorithm measures the minimum number of edits (insertions, deletions, or substitutions) required to transform one string into another. The lower the Levenshtein distance between two strings, the more similar they are. For each slot in a turn, the match slot value is then passed through a fuzzy string matching and multiplied together. The score is then averaged for all turn in the dialogue, which is the metric value itself.

Metrics	Task	Description
UUAF1	User undefined action detection	F1 score dialogue wise, if the prediction consist of at least 1 undefined action, counted as positive
AF1	Action detection	F1 score dialogue wise, counted as a match only when all actions matched
SAF1	Next action prediction	F1 score dialogue wise, counted as a match only when all actions matched
JGA	Slot filling	Accuracy accounted dialogue wise, mean of all turns, calculate using fuzzy string matching

The difference in estimation between UUAF1 and group AF1, SaF1 shown at 1 and 2 respectively.

V. EXPERIMENTS AND RESULTS

1. Result

Model	JGA Seen	JGA Unseen
SDT-seq T5 1.1 XXL	95.8	86.4
AT T5 1.1 Base	89.9	62.4
AT T5 1.1 XXL	94.8	82.2
$\text{IFST}_{Xtags\_Xrand}$ *Base*	77.9	61.1
- Xtags	78.4	63.9
- Xtags_Xrand	77.2	60.7
$\text{IFST}_{\text{Xtags\_Xrand}}$ Large	85.4	72.2
- Xtags	86.6	75.0
- Xtags_Xrand	85.1	72.3

Table 1 JGA ON SGD TEST SET. RESULT OF AT AND SDT-SEQ ARE INFERRED FROM [23] [15]

Despite having a simple dialogue flow, we experimented with SGD and SGD-X. We believe a more complex workflow should be divided into downstream components. We use the flan-T5 series for experiments [26]. flan-T5 differs from previous T5 versions in that it was instruction fine-tuned and able to follow instructions. All models are trained on 4 tasks mentioned above under in-depth fine-tuning (IFST). Models utilize Huggingface trainer API with learning rate 5e-4, batch size 2048, and max step 3000. During in-depth fine tuning, we noticed both validation and training loss drop more significantly with larger batch size. In hypothesis, a larger batch size allows the model to generalize better. We attempt to provide additional prompts for each task, but it is not reflected in the evaluation result.

Model	UUAF1	All AF1	All SaF1 Seen	SaF1 Unseen
AT T5 1.1 Base	-	-	89.8	86.1
AT T5 1.1 XXL	-	-	91.3	88.9
IFST$_{Xtags_Xrand}$ *Base*	85.7	66.3	85.9	82.1
- Xtags	85.4	65.2	82.3	79.6
- Xtags_Xrand	85.6	65.2	81.9	79.5
IFST$_{\mathrm{Xtags_Xrand}}$ Large	93.2	81.2	89.4	88.2
- Xtags	85.7	66.3	85.9	82.1
- Xtags_Xrand	85.5	65.7	86.2	82.2

Table 2 UUAF1, SAF1, AF1 ON SGD TEST SET. RESULTS OF AT ARE INFERRED FROM [23]. NOTICE SAF1 MEASURED IN IFST KEEPS TRACK OF SYSTEM QUERY AND INFORM ACTIONS ONLY

Table II describes our result on action metrics. Notice our SaF1 and AnyTOD SaF1 are quite different as our metric only captures query and informs actions of the system. IFST result, even though lower than state-of-the-art, we offer the mechanic for downstream task operation. IFST achieves high UUAF1, this is the first stage to reach open-domain and multi-domain agents. While our main contribution is the TOD system, our State tracking performance is not as qualified compared to previous works. One conjecture might be due to model sizes which, in theory, are not large enough to have emergent ability [27] [28].

Table 2 and 1 suggest having more explicit tags greatly boost the model's ability to predict next actions in trade-off with a slight reduction of slot filling ability. The result is foreseeable as next actions evaluation consist of only query and inform actions; while new tags particularly created to support query action prediction. Simultaneously, randomizing slot values slightly increase slot filling performance but does not harm action ability. One hypothesis is because these random tokens are not regularly seen during training. Therefore, modification of random tokens does not lead to changes of other general tokens.

VI. CONCLUSIONS AND DISCUSSION

By using symbolic methods, the TOD system can be made more robust, efficient, and interpretable than traditional machine learning-based dialogue systems. IFST gained the ability to define user out-of-scope activities at the sacrifice of little slot-filling performance. We propose a novel framework for the TOD system to work inter-operately with the retrieval augmented system. We also introduce and report the result of the mechanic to track out-of-scope actions for state tracking via action schema.

In this paper, we presented a task-oriented dialog system based on a pre-trained language model. Our system results demonstrate the effectiveness of detecting out-of-scope, while being able to generalize well to new domains, without the need for additional training. There are a few limitations to our study. First, we evaluated our system on the SGD dataset, which may not be representative of all real-world task-oriented dialog systems. Second, our system is still under development, and there is room for improvement in terms of task completion performance and efficiency. In future work, we plan to evaluate our system with SGD in combination with the open domain and provide a broader ablation study of state tracking.

Discussion

In this capstone project, we explored the potential of symbolic dialogue for general domain state tracking (TOD). We developed a novel TOD system that utilizes symbolic methods to effectively track the state of the conversation in task-oriented dialogues. Our system employs the flan-T5 encoder-decoder model, which is fine-tuned on the Schema-Guided Dialogue (SGD) dataset.

Our experiments demonstrated that the proposed TOD system achieves promising results on various evaluation metrics, including joint goal accuracy (JGA), action F1 (AF1), system action F1 (SaF1), and user-undefined action F1 (UUAF1). The system exhibits strong performance in handling out-of-scope actions, indicating its ability to adapt to unexpected situations that deviate from the predefined schema.

The symbolic approach offers several advantages for TOD compared to traditional machine learning-based methods. Symbolic representations provide a more explicit and interpretable representation of the dialogue state, making it easier to reason about the conversation flow and track user intentions. Additionally, symbolic methods are less prone to data biases and can generalize better to unseen situations.

Our work contributes to the advancement of TOD by demonstrating the effectiveness of symbolic methods in this domain. We propose a novel framework for integrating symbolic TOD with retrieval-augmented systems to enhance the overall performance of task-oriented dialogue systems. Furthermore, we introduce a new metric, UUAF1, to evaluate the system's ability to detect and handle out-of-scope actions.

Future Directions

Future research directions include:

Improving state tracking performance: Explore techniques to enhance the system's ability to track complex conversation states and handle ambiguous situations.

Investigating different symbolic representations: Evaluate the effectiveness of alternative symbolic representations, such as first-order logic or semantic graphs, for TOD.

Developing explainable TOD systems: Design methods to make TOD systems more transparent and explainable, enabling users to understand the system's reasoning and decision-making processes.

Integrating TOD with other dialogue system components: Explore ways to integrate TOD with other dialogue system components, such as dialogue policy and natural language generation, to create more cohesive and comprehensive dialogue systems.

Evaluating TOD in real-world applications: Conduct extensive evaluations of TOD systems in real-world scenarios to assess their practical applicability and effectiveness in various user interactions.

By addressing these future directions, we can further advance the development of robust, efficient, and interpretable TOD systems that can effectively support users in achieving their goals through task-oriented dialogues.

VII. REFERENCES

[1] Chien-Sheng Wu, Steven C.H. Hoi, Richard Socher, and Caiming 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)*, pages 917–929, Online, November 2020. Association for Computational Linguistics.

[2] Vahid Noroozi, Yang Zhang, Evelina Bakhturina, and Tomasz Kornuta. A fast and robust bert-based dialogue state tracker for schema-guided dialogue dataset. *CoRR*, abs/2008.12335, 2020.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[4] Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. A simple language model for task-oriented dialogue, 2022.

[5] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, I ˜ nigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasiˇ ´ c. Mul tiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016– 5026, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[6] Wanwei He, Yinpei Dai, Min Yang, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. Space-3: Unified dialog model pre-training for task-oriented dialog understanding and generation, 2022.

[7] Wanwei He, Yinpei Dai, Yinhe Zheng, Yuchuan Wu, Zheng Cao, Dermot Liu, Peng Jiang, Min Yang, Fei Huang, Luo Si, Jian Sun, and Yongbin Li. GALAXY: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection. *CoRR*, abs/2111.14592, 2021.

[8] Adib Mosharrof, M. H. Maqbool, and A. B. Siddique. Zero-shot generalizable end-to-end task-oriented dialog system using context sum marization and domain schema, 2023.

[9] Namo Bang, Jeehyun Lee, and Myoung-Wan Koo. Task-optimized adapters for an end-to-end task-oriented dialogue system, 2023.

[10] Vojt ˇ ech Hude ˇ cek and Ond ˇ rej Du ˇ sek. Are llms all you need for task oriented dialogue?, 2023.

[11] Xinyan Zhao, Bin He, Yasheng Wang, Yitong Li, Fei Mi, Yajiao Liu, Xin Jiang, Qun Liu, and Huanhuan Chen. Unids: A unified dialogue system for chit-chat and task-oriented dialogues, 2021.

[12] Miaoran Li, Baolin Peng, Jianfeng Gao, and Zhu Zhang. Opera: Harmonizing task-oriented dialogs and information seeking experience, 2022.

[13] Zhiyu Chen, Bing Liu, Seungwhan Moon, Chinnadhurai Sankar, Paul Crook, and William Yang Wang. Ketod: Knowledge-enriched task oriented dialogue, 2022.

[14] Jeffrey Zhao, Raghav Gupta, Yuan Cao, Dian Yu, Mingqiu Wang, Harri son Lee, Abhinav Rastogi, Izhak Shafran, and Yonghui Wu. Description driven task-oriented dialog modeling. *CoRR*, abs/2201.08904, 2022.

[15] Raghav Gupta, Harrison Lee, Jeffrey Zhao, Yuan Cao, Abhinav Rastogi, and Yonghui Wu. Show, don't tell: Demonstrations outperform descrip tions for schema-guided task-oriented dialogue. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2022.

[16] Qingyang Wu, James Gung, Raphael Shu, and Yi Zhang. Diacttod: Learning generalizable latent dialogue acts for controllable task-oriented dialogue systems, 2023.

[17] Yuncheng Hua, Xiangyu Xi, Zheng Jiang, Guanwei Zhang, Chaobo Sun, Guanglu Wan, and Wei Ye. Dialog-to-actions: Building task-oriented dialogue system via action-level generation, 2023.

[18] Adam Santoro, Andrew Lampinen, Kory Mathewson, Timothy Lillicrap, and David Raposo. Symbolic behavior in artificial intelligence, 2022.

[19] Vedant Gaur and Nikunj Saunshi. Reasoning in large language models through symbolic math word problems. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5889–5903, Toronto, Canada, July 2023. Association for Computational Linguistics.

[20] Jerry Wei, Le Hou, Andrew Lampinen, Xiangning Chen, Da Huang, Yi Tay, Xinyun Chen, Yifeng Lu, Denny Zhou, Tengyu Ma, and Quoc V. Le. Symbol tuning improves in-context learning in language models, 2023.

[21] Qian Liu, Fan Zhou, Zhengbao Jiang, Longxu Dou, and Min Lin. From zero to hero: Examining the power of symbolic tasks in instruction tuning, 2023.

[22] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[23] Jeffrey Zhao, Yuan Cao, Raghav Gupta, Harrison Lee, Abhinav Rastogi, Mingqiu Wang, Hagen Soltau, Izhak Shafran, and Yonghui Wu. Anytod: A programmable task-oriented dialog system, 2023.

[24] Hamed Zamani, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky. Retrieval-enhanced machine learning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, jul 2022.

[25] Ting Chen, Saurabh Saxena, Lala Li, David J. Fleet, and Geoffrey Hinton. Pix2seq: A language modeling framework for object detection, 2022.

[26] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Sid dhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022.

[27] Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. Larger language models do in-context learning differently, 2023.

[28] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022.

VIII. APPENDIX

1. Domain transition examples

TRANSITION: FLIGHT21

INPUT

[PARAMS] P7=THE TIME OF DEPARTURE FOR THE OUTBOUND FLIGHT; P16=CITY GOING TO; P20=LANDING TIME OF RETURN FLIGHT TO ORIGIN; P24=RETURN LEG FLIGHT DATE; P32=IF THE FLIGHT ARRIVES THE NEXT DAY AS PER DESTINATION LOCAL TIME ;...

[USERACTS] U2=USER INFORM UNDEFINED INFORMATION; ... U264=USER SELECT ITEM; U266=USER INFORM P133; U268=USER THANK

[SYSACTS] S1=INFORM P107; S23=INFORM P53; S24=INFORM P61; S27=INFORM P92; S54=QUERY SEARCHFORROUNDTRIPFLIGHT API; S61=OFFER USER P133; ...

[DEPENDENCIES] ... [TARGETACTS] ...

[CONVERSATION] [USER] I AM TAKING A TRIP, CAN YOU HELP ME RESERVE MY TICKET PLEASE. [SYSTEM] SURE, I WOULD LOVE TO. WHICH CITY WILL YOU BE VISITING? DO YOU KNOW WHAT CITY AND DATE YOU WILL BE DEPARTING FROM? ... [USER] I SEE. I WOULD LIKE TO HEAR MORE OPTIONS. WHAT DOES DELTA AIRLINES HAVE FOR ECONOMY SEATING PRICES? [SYSTEM] DELTA AIRLINES HAS 1 FLIGHT DEPARTING AT 3 PM WITH 1 LAYOVER FOR $328 PER PERSON. [USER] HOLD THAT THOUGHT, LET'S GO AND CHECK MY SAVINGS ACCOUNT BALANCE PLEASE.

OUTPUT

[STATES] P16=LAS VEGAS; P83=NEW YORK; P89=MONDAY NEXT WEEK; P107=ECONOMY; P133=DELTA AIRLINES [HISTORY] U248; S83, S106, S227; U82, U140, U145; S61, S190, S283, S312, S320; U220; S248; U75, U196; S88, S278; U111, U218, U266; S61, S190, S267, S283, S312, S320; U2, U2, U264 [NEXTACTS]

TRANSITION: BANK22

INPUT

[PARAMS] P9=HOW MANY DAYS THE TRANSFER WILL TAKE; P11=THE AMOUNT TO TRANSFER TO THE RECIPIENT; P34=TYPE OF USER'S BANK ACCOUNT: CHECKING OR SAVINGS 34A) CHECKING 34B) SAVINGS; ...

[USERACTS] U5=USER INFORM P11; U17=USER INFORM P34; U29=USER INFORM P114; U30=USER THANK; U40=USER DENY THE OFFER; U47=USER WANT TO GETACCOUNTBALANCE; U59=USER SELECT ITEM; ...

[SYSACTS] S14=REQUEST P11; S47=OFFER USER TRANSFERMONEYTOUSER; S50=OFFER USER P34; S53=REQUEST P34; ... S158=QUERY GETACCOUNTBALANCE API; ...

[DEPENDENCIES] S50, S100, S294, S314, U5, U17, U228 -> S88; U17 -> S158 [TARGETACTS] S158

[CONVERSATION] [USER] I AM TAKING A TRIP, CAN YOU HELP ME RESERVE MY TICKET PLEASE. [SYSTEM] SURE, I WOULD LOVE TO. WHICH CITY WILL YOU BE VISITING? DO YOU KNOW WHAT CITY AND DATE YOU WILL BE DEPARTING FROM? ... [USER] I SEE. I WOULD LIKE TO HEAR MORE OPTIONS. WHAT DOES DELTA AIRLINES HAVE FOR ECONOMY SEATING PRICES? [SYSTEM] DELTA AIRLINES HAS 1 FLIGHT DEPARTING AT 3 PM WITH 1 LAYOVER FOR $328 PER PERSON. [USER] HOLD THAT THOUGHT, LET'S GO AND CHECK MY SAVINGS ACCOUNT BALANCE PLEASE.

OUTPUT

[STATES] P34=SAVINGS [HISTORY] U207; S281, S281, S281; U207, U207, U207; S64, S64, S64, S64; U244; S64; U244, U244; S64, S64; U207, U207, U229; S64, S64, S64, S64, S64; U17, U47, U59; [NEXTACTS] S158 ;

2. *Task breakdown example*

“[params] p9=how many days the transfer will take; p11=the amount to transfer to the recipient; p34=type of user's bank account: checking or savings 34a) checking 34b) savings; p104=amount of money in selected account; p109=name of the person receiving the transfer; p114=type of account to send to 114a) savings 114b) checking

[useracts] u5=user inform p11; u17=user inform p34; u29=user inform p114; u30=user thank; u40=user deny the offer; u47=user want to getaccountbalance; u59=user select item; u86=user goodbye; u123=user want to transfermoneytouser; u207=user inform undefined information; u209=user request p9; u228=user inform p109; u229=user request alternative items; u244=user request undefined information; u257=user agree to the offer

[sysacts] s14=request p11; s47=offer user transfermoneytouser; s50=offer user p34; s53=request p34; s57=ask user if they need anything more; s61=goodbye user; s64=inform undefined information; s88=query transfermoneytouser api; s100=ask to confirm value of p109; s135=notify failure; s158=query getaccountbalance api; s162=ask to confirm value of p114; s241=ask to confirm undefined information; s242=notify success; s252=offer user p104; s272=inform number of items satified user; s279=inform p9; s281=request undefined information; s294=ask to confirm value of p34; s307=request p109; s314=ask to confirm value of p11

[dependencies] s50, s100, s294, s314, u5, u17, u228 -> s88; u17 -> s158 [targetacts] s158

[conversation] [user] i am taking a trip, can you help me reserve my ticket please. [system] sure, i would love to. which city will you be visiting? do you know what city and date you will be departing from? [user] yes, i'm flying new york to las vegas on monday next week. what can you find? [system] if you like american airlines, they have a 6:50 pm with 1 layover for $287 per ticket. [user] would i arrive the same day or the day after? [system] your flight will depart and arrive on the same date. [user] if i go with this flight, where would i be flying out of and where will i arrive? [system] your departure airport is jfk international airport and arrival would be at mccarran international airport. [user] i see. i would like to hear more options. what does delta airlines have for economy seating prices? [system] delta airlines has 1 flight departing at 3 pm with 1 layover for $328 per person. [user] hold that thought, let's go and check my savings account balance please.“

A output example:

“[states] p34=savings

[history] u207; s281, s281, s281; u207, u207, u207; s64, s64, s64, s64; u244; s64; u244, u244; s64, s64; u207, u207, u229; s64, s64, s64, s64, s64; u17, u47, u59

[nextacts] s158”

We break it down to 4 different tasks:

- Slot filling (1)
- User action detection (2)
- System action detection (3)
- Next action prediction (4)

(1)

Input

“[instructions] What are the dialogue state given [params] and [conversation]

[params] p9=how many days the transfer will take; p11=the amount to transfer to the recipient; p34=type of user's bank account: checking or savings 34a) checking 34b) savings; p104=amount of money in selected account; p109=name of the person receiving the transfer; p114=type of account to send to 114a) savings 114b) checking

[useracts] u5=user inform p11; u17=user inform p34; u29=user inform p114; u30=user thank; u40=user deny the offer; u47=user want to getaccountbalance; u59=user select item; u86=user goodbye; u123=user want to transfermoneytouser; u207=user inform undefined information; u209=user request p9; u228=user inform p109; u229=user request alternative items; u244=user request undefined information; u257=user agree to the offer

[conversation] [user] i am taking a trip, can you help me reserve my ticket please. [system] sure, i would love to. which city will you be visiting? do you know what city and date you will be departing from? [user] yes, i'm flying new york to las vegas on monday next week. what can you find? [system] if you like american airlines, they have a 6:50 pm with 1 layover for $287 per ticket. [user] would i arrive the same day or the day after? [system] your flight will depart and arrive on the same date. [user] if i go with this flight, where would i be flying out of and where will i arrive? [system] your departure airport is jfk international airport and arrival would be at mccarran international airport. [user] i see. i would like to hear more options. what does delta airlines have for economy seating prices? [system] delta airlines has 1 flight departing at 3 pm with 1 layover for $328 per person. [user] hold that thought, let's go and check my savings account balance please.“

Output

“[states] p34=savings”

(2)

Input

“[instructions] What are the system last actions given [params], [useracts] and [conversation]

[params] p9=how many days the transfer will take; p11=the amount to transfer to the recipient; p34=type of user's bank account: checking or savings 34a) checking 34b) savings; p104=amount of money in selected account; p109=name of the person receiving the transfer; p114=type of account to send to 114a) savings 114b) checking

[conversation] [user] hold that thought, let's go and check my savings account balance please.“

Output

“[useracts] u17, u47, u59”

(3)

“[instructions] What are the user last actions given [params], [systemacts] and [conversation]

[params] p9=how many days the transfer will take; p11=the amount to transfer to the recipient; p34=type of user's bank account: checking or savings 34a) checking 34b) savings; p104=amount of money in selected account; p109=name of the person receiving the transfer; p114=type of account to send to 114a) savings 114b) checking

[sysacts] s14=request p11; s47=offer user transfermoneytouser; s50=offer user p34; s53=request p34; s57=ask user if they need anything more; s61=goodbye user; s64=inform undefined information; s88=query transfermoneytouser api; s100=ask to

confirm value of p109; s135=notify failure; s158=query getaccountbalance api; s162=ask to confirm value of p114; s241=ask to confirm undefined information; s242=notify success; s252=offer user p104; s272=inform number of items satified user; s279=inform p9; s281=request undefined information; s294=ask to confirm value of p34; s307=request p109; s314=ask to confirm value of p11

[conversation] [system] delta airlines has 1 flight departing at 3 pm with 1 layover for $328 per person.“

A output example:

“[systemacts] s64, s64, s64, s64, s64;”

(4)

“[instructions] What are the next actions system should take in response to the conversation

[params] p9=how many days the transfer will take; p11=the amount to transfer to the recipient; p34=type of user's bank account: checking or savings 34a) checking 34b) savings; p104=amount of money in selected account; p109=name of the person receiving the transfer; p114=type of account to send to 114a) savings 114b) checking

[useracts] u5=user inform p11; u17=user inform p34; u29=user inform p114; u30=user thank; u40=user deny the offer; u47=user want to getaccountbalance; u59=user select item; u86=user goodbye; u123=user want to transfermoneytouser; u207=user inform undefined information; u209=user request p9; u228=user inform p109; u229=user request alternative items; u244=user request undefined information; u257=user agree to the offer

[sysacts] s14=request p11; s47=offer user transfermoneytouser; s50=offer user p34; s53=request p34; s57=ask user if they need anything more; s61=goodbye user; s64=inform undefined information; s88=query transfermoneytouser api; s100=ask to confirm value of p109; s135=notify failure; s158=query getaccountbalance api; s162=ask to confirm value of p114; s241=ask to confirm undefined information; s242=notify success; s252=offer user p104; s272=inform number of items satified user; s279=inform p9; s281=request undefined information; s294=ask to confirm value of p34; s307=request p109; s314=ask to confirm value of p11

[dependencies] s50, s100, s294, s314, u5, u17, u228 -> s88; u17 -> s158 [targetacts] s158

[conversation] [user] i am taking a trip, can you help me reserve my ticket please. [system] sure, i would love to. which city will you be visiting? do you know what city and date you will be departing from? [user] yes, i'm flying new york to las vegas on monday next week. what can you find? [system] if you like american airlines, they have a 6:50 pm with 1 layover for $287 per ticket. [user] would i arrive the same day or the day after? [system] your flight will depart and arrive on the same date. [user] if i go with this flight, where would i be flying out of and where will i arrive? [system] your departure airport is jfk international airport and arrival would be at mccarran international airport. [user] i see. i would like to hear more options. what does delta airlines have for economy seating prices? [system] delta airlines has 1 flight departing at 3 pm with 1 layover for $328 per person. [user] hold that thought, let's go and check my savings account balance please."

A output example:

"[nextacts] s158"