

Statistical Dialogue Systems

Exam questions

NPFL099 – ÚFAL, 2024/2025

Charles University

Statistical dialogue systems	2
Data evaluation	4
Basics of neural networks.....	7
Training neural networks.....	10
Natural language understanding	13
Dialogue state tracking	15
Dialogue policies	17
Natural language generation	20
End-to-end models	22
Chatbots.....	24
Multimodality.....	27
Linguistics and ethics.....	29

Statistical dialogue systems

What's the difference between task-oriented and non-task-oriented systems?

A **dialogue system**, in general, is a computer system designed to interact with users in **spoken natural language**.

- **Task-oriented systems** are designed to **assist users in accomplishing specific tasks**, such as booking a flight, finding restaurants, or setting reminders. These systems often require backend integration to handle structured queries and provide precise results. Examples include smart home assistants and customer service bots.
- **Non-task-oriented systems** are built for **open-ended interactions**, such as social **chatting or entertainment**. They aim to maintain engaging conversations without focusing on completing specific tasks. Examples include chatbots like Xiaolce or systems designed for the Turing test.

Describe the difference between closed-domain, multi-domain, and open-domain systems.

- **Closed-domain systems** handle a single, well-defined area of conversation, such as banking or flight bookings. They perform **highly specialized tasks** within their narrow scope.
- **Multi-domain systems** integrate several closed-domain systems to allow for **conversations across multiple topics**. Virtual assistants like Alexa and Google Assistant, which can handle scheduling, weather updates, and music playback, are examples.
- **Open-domain systems** are designed to respond to any topic, making them capable of engaging in **more general conversations**. These systems, often powered by large language models (LLMs), aim for broad coverage but may lack depth in specific areas.

Describe the difference between user-initiative, mixed-initiative, and system-initiative systems.

- **System-initiative systems** control the flow of dialogue by asking questions and **guiding the user through a structured interaction**. This is the most traditional and robust system, but the least natural. For example, a form-filling process where the system collects information step by step ("*What is your date of birth?*").
- **User-initiative systems** allow the **user to drive the conversation**. The system responds to user queries or commands without steering the interaction. For instance, "*Alexa, play some jazz music*".
- **Mixed-initiative systems** enable **both the system and the user to take turns** leading the conversation. This approach is more natural and flexible but also more complex to implement. An example would be a system that allows a user to interrupt and ask a follow-up question while the system is providing information.

List the main components of a modular task-oriented dialogue system (text/voice-based).

A modular task-oriented dialogue system typically consists of the following components:

- Automatic Speech Recognition (ASR): Converts spoken input into text.
- Natural Language Understanding (NLU): Analyzes the text to extract meaning, such as intents and slot values.
- Dialogue Manager (DM): Decides the next system action based on the dialogue history and user input.
- Natural Language Generation (NLG): Converts the system's intent into natural language responses.
- Text-to-Speech (TTS) / Speech Synthesis: Converts text-based responses into spoken output (for voice-based systems).

What is the task (input/output) of speech recognition in a dialogue system?

The speech recognition module task, in terms of I/O, is:

- Input: Acoustic speech signals (e.g., audio waves).
- Output: Textual representation of the spoken words. This process often includes generating multiple hypotheses with confidence scores (n-best list) to account for uncertainty in recognition, due to potential problems like noise or distance.

What is the task (input/output) of speech synthesis in a dialogue system?

The speech synthesis module task, in terms of I/O, is:

- Input: Textual responses generated by the system (e.g., from NLG).
- Output: Synthetic speech signals that convey the text in spoken form. This includes considerations for pitch, intonation, and prosody to produce natural-sounding speech.

Data evaluation

What are the usual approaches to collecting dialogue data?

- **In-house collection** using experts or students: This method ensures high-quality data but is expensive and time-consuming.
- **Wizard-of-Oz (WoZ)**: A human pretends to be the system, interacting with users to simulate dialogues. It is often used for prototyping and collecting data before a system is fully implemented.
 - o It involves users interacting with what they believe is an automated system, but the system is actually controlled by a **human “wizard.”** This approach enables collecting realistic dialogue data and testing system behavior without having the actual system in place. The wizard typically selects **predefined responses or actions**, which avoids delays associated with free typing.
- **Crowdsourcing**: Involves hiring untrained workers via platforms like Amazon Mechanical Turk to gather large-scale data at lower cost.
- **Web crawling**: Fast and cheap but often results in noisy, non-dialogue-specific data that may require extensive cleaning.

What are the differences between intrinsic and extrinsic evaluation?

- o **Intrinsic evaluation** assesses **components of the system in isolation**, focusing on their **internal performance**, such as the accuracy of NLU or the BLEU score for NLG.
- o **Extrinsic evaluation** measures the **system's effectiveness in its intended real-world application**, such as task success rates in user interactions.

What are the differences between subjective and objective evaluation?

- o **Subjective evaluation** gathers **human opinions** about the system's performance through questionnaires, ratings, or interviews. For example, asking users if the system responses were natural or engaging.
- o **Objective evaluation** relies on **measurable metrics** derived directly from **system outputs**, such as accuracy, precision, recall, or BLEU scores.

What are the common options for getting people to evaluate your system?

- o **In-house evaluations**: Use lab participants, colleagues, or hired testers. This provides controlled, high-quality feedback but is costly and time-intensive.
- o **Crowdsourcing**: Platforms like Amazon Mechanical Turk allow for faster, large-scale evaluations at a lower cost, though data quality might vary.
- o **Real user deployment**: Deploy the system in a real-world setting and analyze interactions. This method provides authentic data but requires more time and resources to attract users.

What are some evaluation metrics for non-task-oriented systems (chatbots)?

- **Engagement metrics:** Number of words per turn or number of questions asked by the chatbot.
- **Coherence:** Semantic similarity between turns, often measured via embedding similarity.
- **Topic breadth and depth:** Average number of distinct topics in a conversation and the length of discussions on specific topics.
- **Self-play metrics:** Sentiment analysis and dialogue sentiment consistence.

How would you evaluate NLU (both slots & intents)?

An intrinsic objective evaluation process for NLU would be as follows:

- **Slot evaluation:** Use precision, recall, and F1-score to measure how accurately the system identifies and extracts slot values.
- **Intent evaluation:** Calculate classification accuracy to assess how well the system detects the user's intent. Multi-class classification metrics may also be used. Also, find exact matches for the whole semantic structure, although we might find lots of ambiguity in user inputs.

Explain an NLG evaluation metric of your choice.

The BLEU (Bilingual Evaluation Understudy) metric is commonly used for NLG. It evaluates the overlap of n-grams (sequences of 1, 2, or more words) between the system's output and one or more reference texts. It uses a brevity penalty to prevent overly short responses, shorter than a given reference, and computes the geometric mean of n-gram precisions.

It is not a really reliable measure, but it has been used for decades now. It is key not to use it over a single sentence, but over a whole dataset, since its correlation with actual, human speech is not really proven.

$$\text{BLEU} = \text{BP} \cdot \exp \left(\sum_{n=1}^4 \frac{1}{4} \log(p_n) \right)$$

range [0,1] (percentage) →

geometric mean

brevity penalty (1 if output longer than reference, goes to 0 if too short)

n-gram precision:
 $p_n = \frac{\sum_u \# \text{ matching n-grams in } u}{\sum_u \# \text{ n-grams in } u}$

Show 2 examples of subjective, human evaluation metrics for dialogue systems.

These metrics often use Likert scales (e.g., 1-5) or continuous sliders for user input

- **Satisfaction score:** "Did you get all the information you needed?"
- **Engagement rating:** "Did you enjoy the conversation with the system?"

What is significance testing, and what is it good for?

Significance testing evaluates whether **observed differences** between systems are **statistically meaningful or could have occurred by chance**, resulting in a given **confidence %** value over a given hypothesis.

For example, it tests the **null hypothesis** (H_0) that two systems perform equivalently. Common methods (with different sensitivity and pre-conditions) include t-tests, Mann-Whitney U tests, and bootstrap resampling.

Why do you need to evaluate on a separate test set?

Evaluating on a **separate test set** ensures that the system's performance reflects its ability to **handle new, unseen data, rather than memorizing patterns from the training set**. This approach **prevents overfitting** and provides a more accurate assessment of **generalization**.

Why you have dialogue systems A and B, and A performs better than B in terms of response BLEU on a dataset of 100 dialogues. Describe how you'd test for significance.

1. Perform a **bootstrap resampling test**: Randomly resample 100 dialogues with replacement multiple times (e.g., 1,000 iterations).
2. Calculate **BLEU scores** for both systems in each iteration.
 - a. Compute the confidence interval of the difference between the scores.
 - b. If the interval excludes 0 with >95% confidence, conclude that A significantly outperforms B.

Basics of neural networks

What's the difference between classification and regression as a machine learning problem?

- **Classification:** The goal is to **predict a categorical output**, assigning an input to one of several predefined classes (e.g., sentiment analysis: positive, negative, or neutral).
- **Regression:** The goal is to **predict a continuous numerical output**, such as predicting the price of a house based on its features.

What's the difference between classification and ranking as a machine learning problem?

- **Classification:** Each input is assigned to **one of several predefined classes** without ordering (e.g., identifying the topic of an email).
- **Ranking:** The goal is to **order** a set of inputs **based on their relevance or priority** for a specific query (e.g., ranking search engine results based on relevance).

What's the difference between sequence labeling and sequence prediction?

- **Sequence labeling:** **Each input in a sequence** is assigned a label (**1-to-1 mapping**), such as tagging each word in a sentence with its part of speech.
- **Sequence prediction / Autoregressive generation:** The model generates an **entire sequence as output**, often with a different length than the input, such as translating a sentence from one language to another.
 - o It uses the current input and previously generated outputs as context. For example, in language generation, the model predicts the next word based on the preceding words, iteratively building the sequence.

What is an embedding, what units can it relate to, and how can you obtain it?

An **embedding** is a **dense vector representation** of discrete items (e.g., words, characters) that captures **semantic or syntactic relationships in the data**. In terms of **units**, it can relate to words, subwords, characters, or even larger structures like sentences.

The ways to obtain them are:

- Random initialization and train as part of a neural network.
- Usage of pretrained embeddings (e.g., Word2Vec, GloVe) trained on large corpora for specific relationships.

What are subwords and why are they useful?

Subwords are smaller units of text derived from breaking words into components, such as prefixes, suffixes, or frequent character sequences. Their usefulness relies on the fact that they:

- Handle out-of-vocabulary words by breaking them into smaller parts.
- Reduce vocabulary size while retaining rich representations.
- Improve performance on morphologically rich languages.

What's an encoder-decoder model and how does it work?

An encoder-decoder model is a neural architecture designed for sequence-to-sequence tasks, often used in tasks like machine translation, parsing and summarization – where more complex structures are linearized into sequences. The way it works is:

1. The encoder processes the input sequence into a fixed-size vector (hidden states). Each next step uses the last hidden state and the following token as input.
2. The decoder generates the output sequence step-by-step, using the encoder's last hidden representation and its own previous outputs.
 - a. Every output of the decoder corresponds with the hidden state and the computation of softmax over output vocabulary + argmax.

How does an attention model work?

Attention mechanisms enhance encoder-decoder models by allowing the decoder to focus on specific parts of the input sequence at each step.

The decoder computes a weighted sum of encoder hidden states (context vector), with the weights determined and changed dynamically based on current, desired relevance. This weighted memory is then fed as decoder inputs and into softmax layer.

Variants include *Bahdanau and Luong attention*, differing in how weights are computed and applied.

What's the main principle of operation for convolutional networks?

Convolutional Neural Networks (CNNs) use convolutional layers to extract features by applying filters (kernels) over input data:

1. A filter slides across the input to compute feature maps via element-wise multiplication and summation.
2. Non-linearities (e.g., *ReLU*) are applied to introduce complexity.
3. Pooling layers reduce dimensions, preserving key features

What's the difference between LSTM/GRU-based and Transformer-based architecture?

- **LSTM (Long Short Term Memory)/GRU (Gated Recurrent Unit):** Sequential processing, using gates to manage long-term and short-term dependencies in data. It is due to sequential nature.
- **Transformer:** Processes sequences in parallel using self-attention mechanisms. It is faster, scalable, and capable of capturing long-range dependencies effectively.

Describe the basics of a Transformer language model.

Transformers are neural architectures that rely entirely on self-attention and feed-forward networks (rather than on encoder gate recurrence), allowing the creation of bigger nets that efficiently are able to process sequences in parallel. The main characteristics are the following:

- **Self-attention:** Computes relationships between all tokens in the sequence simultaneously, focusing on previous decoder steps in order to increase consistency within longer sequences.
- **Positional encoding:** Adds position-dependent information about tokens in the sequence.
- **Layers:** Composed of multi-head attention and feed-forward sublayers, often stacked multiple times.

What's a pretrained language model?

A pretrained language model is a neural network trained on large text corpora to understand language patterns. It acts as a base for downstream tasks, offering "contextual embeddings" that adapt word meanings based on context. Examples are:

- BERT (for classification, tagging).
- GPT (for language generation).
- T5 (for encoder-decoder tasks).

Training neural networks

Describe the principle of stochastic gradient descent.

Stochastic Gradient Descent (SGD) is an optimization algorithm used in supervised training to minimize a cost or loss function. Instead of computing gradients over the entire dataset, which can be computationally expensive, it calculates gradients using random mini-batches of training instances. This approach accelerates convergence and reduces memory requirements, balancing efficiency and effectiveness.

Why is it important to choose an appropriate learning rate?

Choosing the correct learning rate (α) is critical because a value that is too high can cause overshooting, preventing convergence, while a value that is too low may lead to slow progress, making training excessively time-consuming. Proper learning rate selection ensures faster and more stable convergence towards the minimum of the loss function.

Describe an approach for learning rate adjustment of your choice.

One effective approach for adjusting the learning rate is learning rate decay. It starts with a higher learning rate to expedite initial learning and gradually reduces it as the optimization nears convergence. This can be implemented linearly, exponentially, or through stepwise reductions.

The "reduce-on-plateau" strategy lowers the learning rate only when performance improvement stagnates.

What is dropout, what is it good for, and why does it work?

Dropout is a regularization technique where a random subset of neurons is temporarily deactivated (set to zero) during each training iteration. This prevents the network from overfitting to the training data by discouraging co-dependency between neurons. It effectively forces the network to learn more robust and generalized representations of data.

What is a variational autoencoder and how does it differ from a "regular" autoencoder?

A variational autoencoder (VAE) is a probabilistic model that learns to encode data into a latent space defined by distributions (means and variances) rather than fixed point embeddings as in traditional autoencoders. It uses these distributions to generate new data by sampling from them. This approach provides a smoother and more continuous latent space, enabling better generative capabilities.

What is masked language modeling?

Masked Language Modeling (MLM) is a self-supervised pretraining task where some words in a sentence are randomly masked, and the model is trained to predict these hidden words. It improves the model's understanding of context and enhances its ability to generate meaningful embeddings, as seen in BERT-like models.

How do Generative Adversarial Networks work?

Generative Adversarial Networks (GANs) consist of two neural networks that are trained to generate and differentiate believable outputs: a generator and a discriminator. The generator creates synthetic data samples, while the discriminator attempts to differentiate real samples from fake ones.

Through iterative training, the generator improves to the point where its outputs become indistinguishable from real data, while the discriminator becomes better at identifying fakes. The process reaches equilibrium when neither can improve further.

Describe the principle of the pretraining+finetuning approach.

In the pretraining+finetuning paradigm, a model is first pretrained on a large dataset using self-supervised or unsupervised tasks to learn general representations. It is then fine-tuned on a smaller task-specific dataset using supervised learning. This method leverages the model's prior knowledge, enabling it to achieve high performance even with limited task-specific data.

How does clustering work?

Clustering is an unsupervised learning approach that groups data points based on their similarities. Algorithms like k-means partition data into a predefined number of clusters by iteratively adjusting the cluster centroids to minimize within-cluster distances.

Gaussian Mixture Models offer soft clustering by estimating the probabilities of data points belonging to multiple clusters, while hierarchical methods merge or divide clusters to form a tree-like structure.

What is self-supervised training and why is it useful?

Self-supervised training generates pseudo-labels from unlabeled data by leveraging intrinsic structures within the data. Tasks like predicting missing words or denoising corrupted inputs do not require manual annotations, enabling the use of massive datasets.

This process builds versatile representations that can be fine-tuned for various downstream tasks, providing scalability and efficiency.

Describe 2 examples of a self-supervised training task.

One example is next-word prediction, where a model predicts the next word in a sentence, as used in GPT-like models.

Another is **denoising autoencoding**, where a model learns to reconstruct clean data from corrupted inputs, enhancing its ability to handle noisy or incomplete data.

Can you apply a pretrained language model for a new task without finetuning it at all?

Yes, a pretrained language model can be applied without finetuning by employing **prompting techniques**. **Carefully crafted prompts guide the model to perform tasks** like text classification or summarization **based on its pre-learned knowledge**.

How can you finetune a language model without updating all the parameters?

Parameter-efficient finetuning techniques such as **Adapters, LoRA, and Soft Prompts** add small, **task-specific parameters** to the model while **keeping the base parameters fixed**. This approach reduces computational demands and mitigates overfitting, making it suitable for resource-constrained scenarios.

What's instruction tuning and what is it good for?

Instruction tuning adapts language models to follow **explicit task instructions by finetuning** them on diverse **datasets with instruction-style annotations**, which include both instructions in **task description and the solution/target** itself in the prompts. It enhances the model's ability to perform tasks specified by human-readable prompts, improving its usability and alignment with user intent.

Natural language understanding

Design a (sketch) of an NLU neural architecture that joins intent detection and slot tagging.

A common NLU neural architecture for joint intent detection and slot tagging uses a bidirectional encoder such as an LSTM or GRU to process the input sequence – usually, 2 RNN encoders (left-to-right and right-to-left, so you can see everything before you start tagging).

The encoder produces hidden states word-by-word, which are used for slot tagging with a softmax output layer for each word (sequence tagging). For intent detection, the final hidden state (or an attention-weighted summary of all hidden states) is passed to a softmax classifier.

This architecture can also incorporate multi-task training, with cross-connections where slot tagging improves intent detection and vice versa.

Describe language understanding as classification and language understanding as sequence tagging.

- Language understanding as classification treats tasks like intent detection as a multi-class classification problem, where each intent corresponds to a distinct class.
- Language understanding as sequence tagging, on the other hand, frames slot tagging as classifying each word in a sequence with respect to predefined slot categories (e.g., "B-date" or "O"). This allows extraction and understanding of structured information from user utterances, such as entity recognition and labeling.

What is delexicalization and why is it helpful in NLU?

Delexicalization replaces specific slot values, such as named entities, with placeholders that indicate their type (e.g., "restaurant-name"). This reduces the sparsity of the training data by grouping conceptually identical but lexically different entities, simplifying the classification task and improving generalization in NLU systems.

Describe one of the approaches to slot tagging as sequence tagging.

Slot tagging can be approached using sequence tagging techniques like the IOB (Inside-Outside-Beginning) format.

Here, each word is classified as beginning a slot ("B-"), being inside a slot ("I-"), or not being part of any slot ("O"). This approach processes utterances word by word, predicting slot labels while considering their sequential dependencies using models like RNNs, LSTMs, or CRFs. For example, in "Book a flight from Boston," "Boston" may be tagged as "B-departure-city" while other words receive "O" tags.

How can you use pretrained language models (e.g., BERT) for NLU?

Pretrained language models like BERT can be fine-tuned for NLU tasks by adding task-specific heads.

For slot tagging, the final hidden layers are passed through a softmax layer to classify each word into a slot category. For intent detection, the embedding of the [CLS] token can be used in a classification head to identify the intent. BERT enables strong contextual embeddings and supports tasks like sequence tagging and classification without substantial modifications.

How can you combine rules and neural networks in NLU?

Rules can complement neural networks by embedding regex-based features directly into the input or leveraging supervised attentions over these features during training.

Rules can also modify the output, adjusting network predictions using handcrafted constraints. This hybrid approach is especially effective for domains with limited training data or in scenarios requiring domain-specific heuristics.

How can an NLU system deal with noisy ASR output? Propose an example solution.

Noisy ASR output can be addressed using joint Automatic Speech Recognition and NLU models that leverage dual encoders or shared representations trained to optimize end-to-end tasks.

Taking into account also the multiple hypotheses generated by ASR, a solution is combining their predictions in the NLU pipeline, by summing them up and achieving a weighted averaging over intent probabilities.

Dialogue state tracking

What is the point of dialogue state tracking in a dialogue system?

Dialogue state tracking is essential in a dialogue system to maintain a memory of the conversation's history, track user goals, and provide a summarized representation of relevant information. This state is critical for deciding the next system action, ensuring continuity, coherence, and progression towards fulfilling the user's intent.

What is the difference between dialogue state and belief state?

- A dialogue state is a definitive snapshot of what the system assumes about the user's intent, preferences, or goals.
- A belief state is a probabilistic representation that captures uncertainty in the dialogue, aggregating probabilities over possible dialogue states. This allows the system to handle ambiguities and low-confidence inputs by maintaining distributions over potential user intents or slot values

What's the difference between a static and a dynamic state tracker?

- Static state trackers rely on a rule-based or feedforward neural architecture without sequential memory. They use recent observations within a fixed context window and often ignore historical dependencies.
- Dynamic state trackers, such as recurrent neural networks (RNNs), explicitly model dialogue history as a sequence, leveraging dependencies between past and current turns for tracking the dialogue state

What's a partially observable Markov decision process (POMDP)?

A partially observable Markov decision process extends a MDP by assuming the system cannot directly observe the true state of the environment.

Instead, the system receives observations influenced by the true state and maintains a belief state— a probability distribution over all possible states. This framework is used in dialogue systems to manage uncertainty due to ambiguities in user input or speech recognition errors.

Describe a viable architecture for a belief tracker.

One effective architecture for belief tracking is a feedforward or recurrent neural network that processes input features such as NLU outputs, ASR confidence scores, or previous dialogue actions. For instance, a recurrent belief tracker might use LSTMs or GRUs to encode dialogue history and predict probabilities over slot values for the current state. Pretrained language models (e.g., *BERT*) can also be fine-tuned to predict spans or values directly from the text.

MDP:
stochastic
sequential
decisioning

based on
Markov state
transitions,
making
decision
under
uncertainty

What is the difference between state trackers as classifiers vs. as candidate rankers?

- State trackers as classifiers explicitly predict a single state or slot value, typically selecting one outcome from predefined categories. They are often slower but can provide higher accuracy for tasks with fewer values.
- State trackers as candidate rankers, on the other hand, evaluate a smaller subset of potential values, ranking them based on likelihood. Rankers are faster and more scalable, particularly for slots with open-ended or unseen values

Describe the principle of state tracking as span selection.

State tracking as span selection involves using a model to extract values for slots directly from user input, treating it as a span within the text. A common approach is to use pretrained language models like BERT.

The model predicts the start and end positions of the span corresponding to a slot value. This method bypasses the need for enumeration or ranking of values and aligns with reading comprehension tasks, making it suitable for precise and dynamic state tracking

Dialogue policies

Describe the basic reinforcement learning setup (agent, environment, actions, rewards).

Reinforcement learning (RL) involves an **agent** interacting with an **environment**. The agent takes **actions** based on its policy, receives **rewards** as feedback, and **transitions** between states.

The goal is to **learn a policy that maximizes the long-term expected reward** by balancing exploration and exploitation. The RL problem is often modeled as a **Markov Decision Process (MDP)**, which incorporates **state-action-reward dynamics** and considers both immediate and future rewards using a discount factor.

Map the general reinforcement learning setup to a dialogue system situation.

In **dialogue systems**, the agent is the **dialogue manager**, the environment consists of the **user and external systems** (e.g., **databases**), actions represent **system responses** (e.g., providing information, asking for clarification), and rewards are **metrics** such as task success, user satisfaction, or dialogue efficiency.

The dialogue state serves as the **current state in the RL framework**, which evolves based on the agent's actions and user responses.

Why is reinforcement learning preferred over supervised learning for training dialogue managers?

Reinforcement learning is preferred because dialogue systems must optimize across multiple turns, considering **long-term outcomes**. RL's ability to **plan and learn from delayed rewards** makes it more suitable for dynamic and interactive tasks like dialogue management.

Supervised learning, in contrast, provides labels for individual actions **without considering how these actions affect future dialogue states or overall task success**.

What are V and Q functions in a reinforcement learning scenario?

- **V-function** (State-Value Function): Measures the expected return starting from a given **state** under a particular policy.
- **Q-function** (Action-Value Function): Measures the expected return starting from a given state and taking a specific **action under a policy**. It includes the **immediate reward** and the value of subsequent states.

What's the difference between actor and critic methods in reinforcement learning?

- **Actor methods**: Focus on **learning a policy directly by parameterizing it** (e.g., neural networks) and **optimizing the policy using gradients**.

- **Critic methods:** Focus on estimating the value functions (V or Q) and use them to improve the policy.
- **Actor-critic methods:** combine the strengths of actor (policy learning) and critic (value function estimation) approaches. The actor updates the policy using the gradients provided by the critic, which evaluates the policy by estimating value functions, reducing variance and improving learning stability.

Describe a Deep Q Network.

A DQN uses a neural network to approximate the Q-function, which predicts action values. Key features include:

- **Experience replay:** Storing past experiences in a buffer to break correlations and enable efficient batch learning.
- **Target Q-function freezing:** Stabilizing learning by using a fixed Q-function periodically updated.
- **Minibatch updates:** Reducing variance in training by averaging over sampled experiences. like SGD
- **Reward clipping:** Preventing numerical instability by scaling rewards.

Describe the REINFORCE approach.

The **REINFORCE** algorithm is a Monte Carlo policy gradient method that updates policy parameters, using stochastic gradient descent (looping forever), by sampling entire episodes.

This way, it goes by adjusting the policy to increase the probability of actions that lead to high returns, to maximize performance. It is straightforward but can suffer from high variance.

What's the main principled difference between Deep Q-Networks and Policy Gradient methods?

DQN focuses on approximating the Q-function and indirectly optimizing the policy, whereas policy gradient methods directly optimize the policy by parameterizing and adjusting it based on observed returns. Policy gradients can better handle high-dimensional, continuous action spaces.

What's the difference between on-policy and off-policy optimization?

- **On-policy:** The policy being learned is the same as the one generating the data (e.g., REINFORCE).
- **Off-policy:** The learning policy differs from the data-generating policy, allowing for reuse of data from different policies (e.g., Q-learning, ACER).

Why do you typically need a user simulator to train a reinforcement learning dialogue policy?

A **user simulator** replicates user behavior for RL training, providing a scalable and controlled environment for training and evaluation. Approaches include:

- Rule-based simulators: Handcrafted rules to simulate user actions.
- Statistical models: Probabilistic models trained on user interaction data.
- Large Language Models (LLMs): Generate realistic responses and evaluate dialogue quality.

Training dialogue systems with RL requires a large number of interactions, which are impractical with real users.

Give an example of possible turn-level or dialogue-level rewards for RL optimization.

- **Turn-level rewards**: Based on **individual dialogue turns**, such as minimizing repetition or optimizing response relevance.
- **Dialogue-level rewards**: Evaluated at the **end of the interaction**, such as task success, user satisfaction, or completion time.

Natural language generation

What are the main steps of a traditional NLG pipeline?

A traditional natural language generation pipeline typically includes the following steps:

1. Content Planning: Deciding what information to include in the output. This step involves selecting and organizing content to meet the communication goal.
2. Surface Realization: Transforming the planned content into grammatically correct and fluent text. This includes word choice, linearization, and ensuring proper syntax and morphology.

Describe a chosen handcrafted approach to NLG.

One common handcrafted approach is template-based NLG, where pre-defined templates are filled with data. For example, a weather reporting system might use the template: "The temperature in [city] is [temperature] degrees." This approach is straightforward and ensures grammatical correctness but lacks flexibility and scalability.

What are the main problems with vanilla neural NLG systems (base RNN/Transformer)?

Vanilla neural NLG systems, such as those based on RNNs or basic Transformers, suffer from several issues:

- Hallucination: Generating content that is not supported by the input data.
- Semantic Inaccuracies: Omitting or misrepresenting critical details.
- Low Diversity: Repetitive or overly generic responses.
- Data Dependency: Requiring large, high-quality datasets for training.

What is a copy mechanism/pointer network?

A copy mechanism allows the model to select between generating a new token from the vocabulary or copying a token from the input. This is especially useful in tasks involving structured data or low-resource scenarios, where the model must preserve specific terminology or entities from the input.

What is delexicalization and why is it helpful in NLG?

Delexicalization replaces specific entities (e.g., names, locations) with placeholders during training (e.g., <name>). This reduces data sparsity and helps the model generalize better to unseen data. At inference, placeholders are replaced with context-appropriate values.

Describe a possible neural approach to NLG with an approach to combat hallucination.

A neural approach to combat **hallucination** involves **reranking with pretrained language models (PLMs)**. After generating multiple candidate outputs, a reranker assesses their **semantic alignment with the input and selects the most accurate one**. Additionally, fine-tuning PLMs on cleaned or domain-specific data reduces hallucinations.

How can you reduce NLG hallucination by altering the training data?

Training data can be altered to **reduce hallucinations** by:

- **Data Cleaning**: Removing noisy or inconsistent examples.
- **Paraphrasing**: Increasing data diversity by creating varied expressions of the same content.
- **Task-Specific Augmentation**: Adding synthetic data that reflects domain-specific nuances.

How can you use pretrained language models in NLG?

Pretrained language models can be used for NLG by:

- **Fine-Tuning**: Adapting the model to domain-specific tasks using supervised training on task-specific data.
- **Prompting**: Utilizing carefully crafted prompts to guide the model's output without additional training, as seen with large language models like GPT-3.

How can you combine templates with neural approaches for NLG?

Templates can be combined with neural approaches by **generating initial drafts using templates and then refining (post-editing) them with a neural model**. This hybrid approach ensures controllability from templates and fluency from neural networks.

What are the typical decoding approaches in NLG?

Two common **decoding** approaches are:

1. **Greedy Search**: At each time step, selects the **most probable token**. This is computationally efficient but can lead to suboptimal, dull responses due to its inability to consider alternative sequences.
2. **Beam Search**: Explores **multiple hypotheses by keeping the top k sequences** at each step. This provides better approximations of the most probable sequence but is more computationally intensive. Beam search generally produces more coherent and diverse outputs than greedy search.

End-to-end models

What are some pros and cons of end-to-end models over traditional modular ones?

End-to-end models combine the entire dialogue system into one neural network, which enables joint optimization through backpropagation if the model is fully differentiable. This approach reduces the issue of error propagation between components and may improve the system's performance holistically. End-to-end systems often still retain some compositional elements and may require dialogue act-level annotation.

In contrast, traditional modular systems are more flexible, allowing individual components to be improved or replaced independently. However, error accumulation is a significant drawback, and enhancements to one component may not necessarily improve the system overall.

sequence/autoregressive generation

Describe an example structure of an end-to-end dialogue system.

An example of an end-to-end system involves training a sequence-to-sequence (seq2seq) model with components like an encoder and decoder.

For instance, the system may encode previous dialogue states, system responses, and the user's input, then decode the dialogue state before generating a system response.

A database (DB) lookup is typically performed using an external module, and the query results are integrated into the generation process.

Describe the Sequicity (2-step decoding) model.

Sequicity is a simplified seq2seq-based model where the decoder first generates the dialogue state and then generates the system response. It employs a pointer-generator mechanism to handle specific tasks like copying from the dialogue history or database results.

last version
of lora model

This model explicitly maintains a dialogue state and uses a database query with a one-hot vector to indicate the number of results, enabling the system to condition the generation process accordingly.

Describe an end-to-end model based on pretrained language models.

An end-to-end model like the one based on GPT-2 adapts the pretrained language model by introducing specific embeddings for system and user inputs. GPT-2 operates as a decoder-only model where input is passed through all layers during encoding, but predictions are ignored. Training involves supervised generation and classification tasks without belief tracking or database updates. The model uses gold-standard belief states and database outputs for generation.

> user/system
embeddings as inputs
> prompt finetuning

How would you adapt a pretrained language model for an end-to-end dialogue system?

To adapt a pretrained language model, domain-specific embeddings are added for distinguishing between user and system inputs. The model can be fine-tuned on dialogue datasets with history, state, and database results formatted as sequential inputs.

Further customization includes tasks like explicit system action decoding or multi-task training to classify generated responses. Techniques like data augmentation and domain-specific training further enhance performance.

What are “soft” database lookups in end-to-end dialogue systems?

Soft database lookups involve incorporating database interaction directly into the neural network. Systems like Mem2Seq treat database tuples and dialogue history as part of the input memory and use attention mechanisms during decoding to select values from the database. This approach ensures the entire process is differentiable, enabling backpropagation through the database interaction steps.

How would you use reinforcement learning to train an end-to-end model?

Reinforcement learning (RL) for end-to-end models involves rewarding the system based on task success or other criteria like fluency. RL can learn directly from user interactions and optimize the model holistically. However, training entirely with RL on a word level is challenging because it often disregards linguistic naturalness and focuses solely on task completion, which leads to disfluent outputs.

Why is it a bad idea to train end-to-end dialogue systems only with reinforcement learning on word level?

Training solely with RL on the word level can result in unnatural language outputs since RL optimizes task success metrics rather than fluency or coherence. This limitation necessitates combining supervised pretraining with RL or introducing fluency rewards to ensure the outputs remain natural and user-friendly.

How can you increase consistency (w.r.t. dialogue state) in an end-to-end model?

Consistency in dialogue states can be improved through methods like multi-task training, where the model simultaneously generates responses and classifies state consistency. For example, models like *SOLOIST* or *AuGPT* include tasks where artificially corrupted states or responses are used to train the system to detect inconsistencies. Decoding methods like generating state differences (Levenshtein states) also help maintain dialogue state consistency across interactions.

Chatbots

What are the three main approaches to building chatbots?

The three main approaches to building chatbots are:

- **Rule-based chatbots** rely on predefined scripts and keyword spotting to produce responses, which makes them straightforward but inflexible and labor-intensive to create.
- **Retrieval-based chatbots** work by searching a database of pre-existing dialogues to find and rank the most relevant responses for user queries, offering consistent and fluent replies but lacking the ability to generate novel outputs.
- **Generative models**, like seq2seq architectures, create responses from scratch by training on dialogue datasets, which enables handling unseen inputs and generating original replies but often suffers from coherence and diversity issues.

How does the Turing test work? Does it have any weaknesses?

The Turing Test evaluates a machine's ability to exhibit human-like behavior in a conversation by having an evaluator interact with both a human and a machine, identifying which is which. If the evaluator cannot reliably distinguish between the two, the machine is considered to have passed the test.

However, the test has several weaknesses: it focuses solely on mimicking human interaction rather than actual intelligence or understanding, making it susceptible to being "gamed" by superficial tricks like evasive answers or overuse of generic responses. It also fails to account for non-verbal or task-specific aspects of intelligence.

What are some techniques rule-based chatbots use to convince their users that they're human-like?

Rule-based chatbots employ techniques like pattern-matching rules, precedence systems to prioritize certain responses, and fallback options such as generic statements ("I see, please go on"). They mimic human behavior by reformulating user phrases, signaling understanding through repetition, and framing their role (e.g., as a therapist). The simplicity and predictability of these roles make it easier to create the illusion of human-like interaction.

Describe how a retrieval-based chatbot works.

A retrieval-based chatbot functions by maintaining a large corpus of pre-existing dialogues and operating in two main steps.

First, it retrieves a set of potential responses based on the similarity between the user's input and the entries in the corpus, often using methods like TF-IDF.

Second, it reranks these candidates using more sophisticated models, such as neural networks, to select the most contextually appropriate response.

This approach ensures consistent replies but cannot generate unseen sentences, which may limit flexibility.

Why is plain seq2seq-based architecture for chatbots problematic?

Plain seq2seq architectures for chatbots often produce generic and repetitive responses due to the high likelihood of "safe" outputs under maximum likelihood estimation (MLE).

They struggle with encoding and utilizing long contexts, leading to inconsistent or contradictory replies. These architectures also lack an inherent mechanism for maintaining a consistent personality or understanding the conversation's semantic nuances, resulting in incoherent or contextually irrelevant outputs.

Describe an example approach to improving diversity or coherence in a seq2seq-based chatbot.

These techniques address the shortcomings of seq2seq models by either improving their output quality or making better use of conversational context.

One approach to improving diversity is mutual information maximization (MML), which penalizes dull responses by optimizing for a trade-off between response relevance and informativeness.

Another method to enhance coherence involves hierarchical recurrent encoder-decoder (HRED) models, which include an additional turn-level encoder to better handle longer contexts.

How can you use a pretrained language model in a chatbot?

Pretrained language models like GPT-2 or BlenderBot can be adapted for chatbots by fine-tuning them on dialogue-specific datasets. These models can also incorporate domain-specific embeddings or persona embeddings to personalize responses.

Retrieval-augmented generation or multi-task training with response ranking and classification further enhances their conversational quality. Pretrained models leverage their large-scale training to provide fluent and contextually rich replies.

Describe a possible architecture of a hybrid/ensemble chatbot.

A hybrid chatbot combines multiple techniques to provide a more versatile conversational experience. It may use rule-based systems for sensitive or frequently asked questions, retrieval-based methods for factual responses like trivia, and generative models for open-ended dialogues.

A central dialogue management system selects the appropriate method based on the context or topic, often incorporating a topic switch detector or ranking system to refine responses. This architecture is commonly seen in advanced conversational systems like those competing in the Alexa Prize.

How does retrieval-augmented generation work?

Retrieval-augmented generation combines retrieval and generation techniques. It first retrieves a candidate response or relevant information from a database or corpus, which is then edited or enhanced by a seq2seq model to better align with the conversation context.

This approach uses mechanisms like α -blending to balance between copying retrieved content and generating new content. It improves the chatbot's ability to produce informed, contextually appropriate, and coherent responses while leveraging existing knowledge bases.

Multimodality

How does the structure of (traditional) multimodal dialogue systems differ from non-multimodal ones?

Traditional multimodal dialogue systems extend the basic structure of voice/text-based dialogue systems by incorporating multiple input and output modalities. Each modality has its dedicated processing modules akin to NLU (natural language understanding) and NLG (natural language generation). These modules handle inputs like gestures, touch, or gaze and outputs like graphics or body movements.

Unlike non-multimodal systems, these systems integrate and merge interpretations from different modalities and distribute the dialogue manager's output across the various output modules, making the architecture more complex.

Give an example of 3 alternative input modalities (i.e. not voice/text).

- Touch input, including active (e.g., swiping) and passive (e.g., pressing).
- Gesture recognition, such as hand or body movements.
- Gaze tracking, which detects and interprets where the user is looking.

Give an example of 3 alternative output modalities (i.e. not voice/text).

- Graphics, such as maps or visual cards.
- Body movements, including gestures performed by virtual agents or robots.
- Facial expressions, used to convey emotions or engagement in virtual agents

How would you build a multimodal end-to-end neural dialogue system (e.g. for visual dialogue)?

To create a multimodal end-to-end neural dialogue system:

1. Start with a base architecture like a transformer or HRED.
2. For input: use pretrained image recognition models (e.g., ResNet or Faster R-CNN) for visual input processing and pretrained language models for text inputs. These components are integrated using cross-attention mechanisms to align visual and textual features.
3. For output: the system generates text responses and may rank images or gestures based on the multimodal context.
4. Pretraining and fine-tuning on multimodal datasets ensure task-specific optimization.

Explain some problems that may occur when a dialogue system talks to two people at once.

When interacting with multiple users simultaneously, dialogue systems face challenges like identifying who is speaking, ensuring accurate speaker diarization, and maintaining separate conversation contexts.

Gaze or engagement tracking can help identify intended addressees, but ambiguity may arise in determining which user a response should target. Overlapping dialogues or conflicting user inputs can further complicate the interaction

What's the difference between image classification and object detection?

- Image classification assigns a single label to an entire image, indicating the primary object or category it contains (e.g., "cat" or "car").
- Object detection identifies and localizes multiple objects within an image by predicting their bounding boxes and classifying each detected object. This makes object detection more complex than image classification.

How would you build a neural end-to-end image-based dialogue system (consider using pretrained components)?

An image-based dialogue system can use pretrained convolutional neural networks like ResNet or Faster R-CNN for image feature extraction.

These features are combined with textual inputs using cross-modal attention layers.

Transformers or memory networks manage the dialogue context, integrating image and text representations for generating or selecting responses. Fine-tuning on visual dialogue datasets enhances the model's capability to reason about image-based queries.

What is the task of visual dialogue about?

The task of visual dialogue involves meaningful multi-turn interactions centered on an image. A user asks questions about an image, which only the system sees, while the user relies on textual captions or prior exchanges. The system must handle follow-up questions requiring contextual understanding, coreference resolution, and reasoning about the image's content. While it primarily serves as a benchmark for evaluating multimodal systems, it is less practical outside research settings.

Linguistics and ethics

1.1 What are turn taking cues/hints in dialogue?

Turn-taking cues/hints are signals that participants in a dialogue use to determine when one speaker's turn (continuous utterance) ends, and another's begins. These cues ensure smooth communication, minimizing overlaps, ambiguity and gaps. Examples include:

- Linguistic cues: finishing sentences or change in voice pitch.
- Timing cues: silent gaps that indicate a turn is over.
- Non-verbal cues: Eye contact or gestures that show readiness to listen or speak

1.2 What is a barge-in?

A barge-in happens when a speaker interrupts or starts speaking during another speaker's turn. This can occur naturally in conversations, especially in moments of excitement or urgency.

In dialogue systems, barge-ins are a feature allowing users to interrupt an ongoing system response to quickly provide new input.

1.3 What is grounding in dialogue?

Grounding is the process of establishing mutual understanding in a conversation. It ensures that dialogue participants share and, knowingly so, confirm their understanding of the information being exchanged.

This confirmation is given via grounding signals, where the speaker presents and utterance and the listener accepts it by providing evidence of understanding.

This involves creating a common ground, a set of shared knowledge and assumptions that both participants acknowledge. Grounding is particularly important in tasks where precise communication is critical, such as confirming a flight reservation or verifying information.

1.4 Give some examples of grounding signals in dialogue.

Grounding signals help validate understanding of the establishment of a common ground, or indicate confusion. These can either be positive, or negative:

- Positive grounding signals:
 - Visual cues like nodding, smiling, or maintaining eye contact.
 - Backchannels like "uh-huh", "yeah", or "I see" to show active listening.
 - Explicit confirmations like "Yes, I understand."
 - Implicit confirmations through relevant responses in follow-up turns.
- Negative grounding signals:
 - Puzzled expressions or a lack of response.

- Clarification requests such as *"What do you mean?"*.
- Repair requests to correct misunderstandings, like *"Did you mean X?"*.

1.5 What is entrainment/alignment/adaptation in dialogue? What is entrainment/alignment/adaptation in dialogue?

Entrainment refers to how people subconsciously adapt their speech and language behavior to their dialogue partner over the course of a conversation. This can involve:

- Lexical alignment: Using similar words or phrases.
- Syntactic alignment: Adopting similar sentence structures.
- Prosodic alignment: Matching speech rate, pitch, or loudness.
- Accent adaptation: Shifting accents or dialect features to align with the partner.

These phenomena make the dialogue feel more natural, which also helps with social bonding and mutual understanding. While systems typically don't entrain (as it is based on templates and lack of actual, natural context), people entrain to them.

1.6 Describe the overgeneralization/overconfidence problem in data-driven NLP models.

Overgeneralization and overconfidence occur when a model incorrectly assumes high certainty in its outputs, even when the information provided is inaccurate. This is often due to the inherent design of language models, which prioritize generating answers everytime, over acknowledging uncertainty.

For example, a model might confidently give incorrect medical advice because it lacks the ability to assess when it should answer with *"I don't know"*. Potential solutions include training models to be able to answer this (but also making sure that they do not over-use it), or using data augmentation to improve robustness.

1.7 Describe the demographic bias problem in data-driven NLP models.

Demographic bias arises when NLP models fail to represent or perform equally well across all demographic groups. This typically occurs because the training data predominantly reflects dominant or well-documented populations, such as white males in English-speaking countries.

The effects of such bias include reinforcing stereotypes, unequal access to services, and diminished usability for underrepresented groups. For instance, voice assistants may struggle to recognize accents or dialects that deviate from the dominant training set.

1.8 Give an example of a user safety concern in dialogue systems.

A significant **user safety concern** is the handling of **sensitive topics**, such as mental health or emergencies. For instance, a chatbot discussing suicide may provide advice that is misleading or harmful, potentially worsening the user's condition.

This is especially critical in **healthcare-related systems**, where incorrect or insensitive responses could have severe consequences. Another safety concern involves in-car systems, where startling responses or excessive cognitive load can distract the driver, leading to accidents.

1.9 What is the problem with training neural models on private data?

Training neural models on **private data** poses significant risks to **user privacy**. Such models can inadvertently memorize, record and **leak sensitive information**, such as passwords, addresses, or private conversations, when prompted in specific ways.

This can be checked by executing synthetic experiments where we train a seq2seq model with dialogue data and passwords. **By providing the same context, we can retrieve the password that was said.**

For example, researchers have shown that models trained on dialogue data can leak private information embedded in training examples. Voice systems sometimes **record parts of conversations by accident**, which might include sensitive data – which can then be checked by actual humans. This risk is compounded by the fact that some voice assistant systems store and review user interactions, raising ethical concerns about data misuse