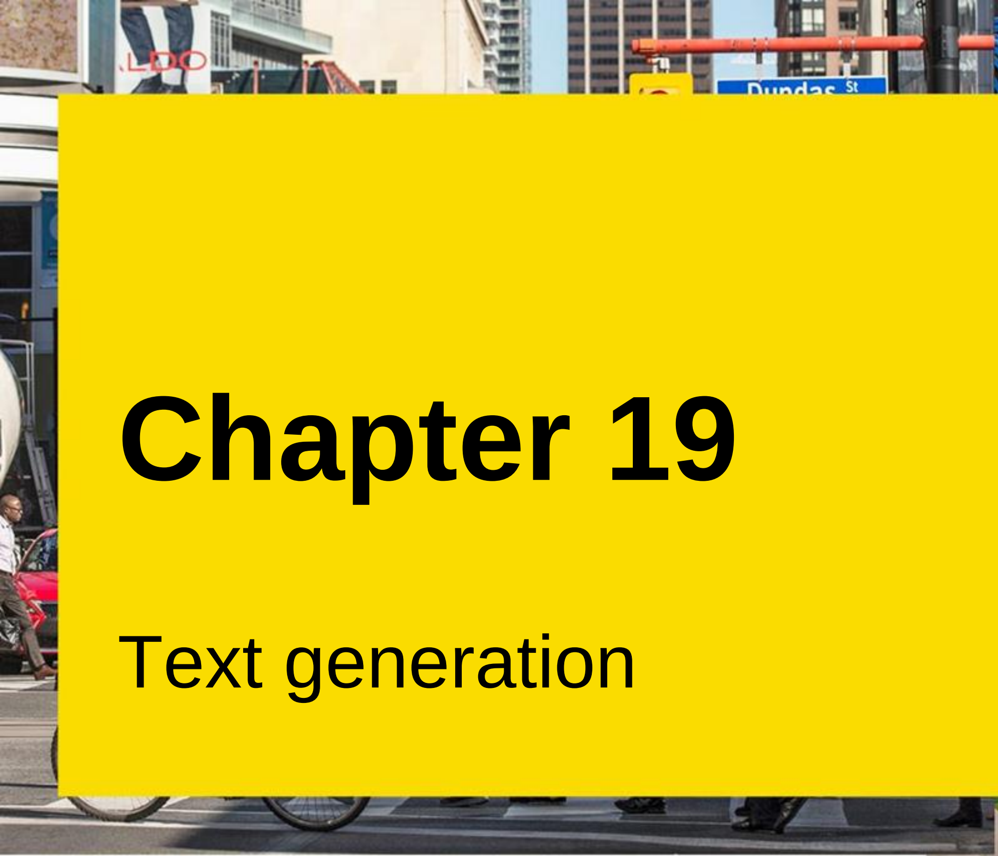


Chapter 19

Text generation

Ryerson University

This chapter …

- This chapter emphasizes **three** main scenarios:
- **Data-to-Text**, in which text is generated to explain or describe a **structured** record or **unstructured** perceptual input;
- **Text-to-Text**, which typically involves fusing information from **multiple** linguistic sources into a **single** coherent summary;
- **Dialogue**, in which text is generated as part of an interactive conversation with one or more human participants.

Ryerson University

Data-to-text generation

- Input : Structured records + Unstructured data
- Structured records **example** : Description of a weather forecast
- Unstructured perceptual data **example** **: R**aw image or video

Ryerson University

Data-to-text generation steps

- All data-to-text systems share some of the same challenges.
  - Determining **what parts** of the data to **describe**;
  - Planning a **presentation** of this information;
  - **Lexicalizing** the data into words and phrases;
  - Organizing words and phrases into well-formed **sentences** and **paragraphs**.
- The earlier stages of this process are sometimes called **content selection** and **text planning**; the later stages are often called **surface realization**.

Example:

- An example input-output pair for the task of **generating text** descriptions of **weather forecasts**

Input

Temperature

time	*min*	*mean*	*max*
06:00-21:00	9	15	21

Cloud sky cover

time	*percent (%)*
06:00-09:00	25-50
09:00-12:00	50-75

Wind speed

time	*min*	*mean*	*max*
06:00-21:00	15	20	30

Wind direction

time	*mode*
06:00-21:00	S

Output

Cloudy, with temperatures between 10 and 20 degrees. South wind around 20 mph.

Ryerson University

Example

- Surface realization can be performed by **grammars** or **templates**, which link specific types of data to candidate words and phrases.
- Example template for generating **descriptions** of **basketball games**

Input: The <team1> (<wins1>-losses1) defeated the <team2> (<wins2>-<losses2>), <pts1>-<pts2>.

Output: The New York Knicks (45-5) defeated the Boston Celtics (11-38), 115-79.

For more **complex** cases, it may be necessary to apply **morphological inflections** such as pluralization and tense marking

Data-to-text generation

- More recent systems are unified **models** that **are trained end-to-end using backpropagation**.
- Problem of alignment: labeled examples provide the data and the text, but they do not specify **which parts of the text correspond to which parts of the data.**
- Solution: both **latent variables** and **neural attention** have been proposed as solutions

Ryerson University

Latent data-to-text

- Consider given a **dataset of texts** and **associated records** $\boxed{\{(\boldsymbol{w}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N}}$
- The goal is to learn a model Ψ, so that

$$\boxed{\hat{\boldsymbol{w}} = \underset{\boldsymbol{w} \in \mathcal{V}^*}{\operatorname{argmax}} \Psi(\boldsymbol{w}, \boldsymbol{y}; \boldsymbol{\theta})}$$

- $\mathcal{V}^*$ is the set of strings over a discrete vocabulary
- $\boldsymbol{\theta}$ is a vector of parameters
- Relationship between $\boldsymbol{w}$ and $\boldsymbol{y}$: the data $\boldsymbol{y}$ may contain dozens of records, and $\boldsymbol{w}$ may extend to several sentences

Latent data-to-text alignment

$$\{(\boldsymbol{w}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{z}^{(i)})\}_{i=1}^{N}$$

- Let's decompose the **scoring function** Ψ, into subcomponents.
- Consider if given an **alignment**, specifies which element of $\boldsymbol{y}$ is expressed in each part of $\boldsymbol{w}$, specifically, let z_m indicates the record aligned to word m.

$$\Psi(\boldsymbol{w}, \boldsymbol{y}; \boldsymbol{\theta}) = \sum_{m=1}^{M} \psi_{w,y}(\boldsymbol{w}_m, \boldsymbol{y}_{z_m}) + \psi_w(w_m, w_{m-1}) + \psi_z(z_m, z_{m-1}).$$

- Given an observed set of **alignments**, the score for a generation can be written as sum of local scores

Ryerson University

Latent data-to-text alignment

$$\Psi(\boldsymbol{w}, \boldsymbol{y}; \boldsymbol{\theta}) = \sum_{m=1}^{M} \psi_{w,y}(\boldsymbol{w}_m, \boldsymbol{y}_{z_m}) + \psi_w(w_m, w_{m-1}) + \psi_z(z_m, z_{m-1})$$

- Given an observed set of alignments, the score for a generation can be written as sum of local scores
- ψ_w can represent a **bigram language model**
- ψ_z can be tuned to **reward coherence**, such as the use of related records in nearby words.
- The parameters of this model could be learned from labeled data:

$$\{(\boldsymbol{w}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{z}^{(i)})\}_{i=1}^{N}$$

Ryerson University

Latent data-to-text alignment

- **Problem:** The alignments between text and records are usually not available.
- **Solution:** solution is to **model the problem probabilistically and** treating the alignment as a **latent variable**.
- The model can then be estimated using expectation maximization or sampling

Neural data-to-text generation

- The **encoder-decoder model** and **neural attention** can be applied to **data-to-text** generation, with the data acting as the source language in machine translation.

- In data-to-text generation, the **attention mechanism** can **link** each part of the **generated text** back to a **record** in the data.

Data encoders: discrete sets

- In some types of **structured records**, all values are drawn from **discrete sets**.

  Example : The birthplace of a person is from a **discrete** set of locations

- In such cases, **vector embeddings** can be estimated for each field and possible value

  Example : One **vector embedding** for the field *BIRTHPLACE*, and another embedding for the value *BERKELEY CALIFORNIA*

- The **table of such embeddings** serves as the **encoding of structured record**
- It is also possible to **compress** the entire table into a **single vector** representation, by **pooling** across the embeddings of each field and value

Ryerson University

Data encoders : Sequences

- Some types of **structured records** have a **natural ordering**.
- We can resemble this **natural ordering** as **series of events** in a game
- Each **event** is a single record, and **can be encoded** by a **concatenation** of vector representations for the **event type**, the **event field**, and the **event values**
- This encoding can then act as the **input layer** for a **recurrent neural network**, yielding a **sequence of vector representations**
- This sequence-based approach can work even in cases where there is no natural ordering over the records

Ryerson University

Sequences : robot soccer match Example

- The following records describe a **sequence of events** in **a robot soccer match**.

$$\text{PASS}(\text{arg1} = \text{PURPLE6}, \text{arg2} = \text{PURPLE3})$$
$$\text{KICK}(\text{arg1} = \text{PURPLE3})$$
$$\text{BADPASS}(\text{arg1} = \text{PURPLE3}, \text{arg2} = \text{PINK9}).$$

- Each **event** is a **single record**, and can be encoded by a concatenation of vector representations for the event type (PASS), the event field (arg1), and the event values (PURPLE3),

$$\mathbf{X} = \left[\boldsymbol{u}_{\text{PASS}}, \boldsymbol{u}_{\text{arg1}}, \boldsymbol{u}_{\text{PURPLE6}}, \boldsymbol{u}_{\text{arg2}}, \boldsymbol{u}_{\text{PURPLE3}}\right].$$

- This encoding can then act as the input layer for a RNN

Data encoders : Images

- The data-to-text goal for images is the **generation of text captions for images**

- Images are represented as **tensors**: A color image of 320×240 pixels would be stored as a tensor with $320 \times 240 \times 3$ intensity values.

- **Dominant** approach to image classification is to **encode images as vectors** using a **combination** of **convolution** and **pooling**

- **Alternative** approach is we can **apply a set of convolutional networks**, yielding **vector** representations for **different parts of the image**, which can then be **combined** using **neural attention**

Data encoders : Dominant Approach

In **dominant** , we **encode images as vectors** using a **combination** of **convolution** and **pooling**

- The **convolution** is applied across the vertical, horizontal, and color dimensions for images.
- By **pooling** the results of successive convolutions, the image is converted to a **vector representation**
- This **vector representation**, can then be fed into the **decoder** as the initial state similar to **sequence-to-sequence** translation mode

Ryerson University

Data encoders : Alternative Approach

- Given a set of **CNN embeddings** of the data $\{z_r\}_{r=1}^{R}$, a **decoder state** h_m, an **attention vector** over the data can be computed.

- When generating word m of the output, attention is computed over the records

$$\begin{aligned}
\psi_\alpha(m,r) &= \boldsymbol{\beta}_\alpha \cdot f(\Theta_\alpha[\boldsymbol{h}_m; \boldsymbol{z}_r]) \\
\boldsymbol{\alpha}_m &= g\left([\psi_\alpha(m,1), \psi_\alpha(m,2), \ldots, \psi_\alpha(m,R)]\right) \\
\boldsymbol{c}_m &= \sum_{r=1}^{R} \alpha_{m\to r} \boldsymbol{z}_r,
\end{aligned}$$

- f is an elementwise nonlinearity such as $\tanh$ or ReLU,
- g is a either **SoftMax** or **elementwise Sigmoid**.
- The weighted sum $\boldsymbol{c}_m$ gets included in the recurrent update to the decoder state, or in the emission probabilities

Ryerson University

A **CNN** gets applied to a set of image locations, and the output at each location ℓ is represented with a vector $\boldsymbol{z}_\ell$ Attention then can then be computed over the image locations.

Neural attention in Image captioning

- Examples of the image captioning task, with attention masks shown for each of the underlined words

A <u>dog</u> is standing on a hardwood floor.

A <u>stop</u> sign is on a road with a mountain in the background.

A **CNN** gets applied to a set of image locations, and the output at each location ℓ is represented with a vector z_ℓ Attention then can then be computed over the image locations.

Ryerson University

Decoder

- Given the encoding, the decoder can function similar to in neural machine translation
- The decoder works using the **attention-weighted encoder representation** in the decoder recurrence and/or output computation

- **Problem :** Sometimes we need to generate words that **do not appear** in the **training vocabulary**;

- **Solution :** Such tokens can be generated in the text by **copying** them over from the input

Gulcehre et al., 2016

Generation and Copy mechanism

- Let's introduce a **variable** $s_m \in \{\text{gen}, \text{copy}\}$, indicating whether token $w_m^{(t)}$ should be **generated** or **copied**. The **decoder probability** is then:

$$\mathrm{p}(w^{(t)} \mid \boldsymbol{w}_{1:m-1}^{(t)}, \mathbf{Z}, s_m) = \begin{cases} \mathrm{SoftMax}(\boldsymbol{\beta}_{w^{(t)}} \cdot \boldsymbol{h}_{m-1}^{(t)}), & s_m = \text{gen} \\ \sum_{r=1}^{R} \delta\left(w_r^{(s)} = w^{(t)}\right) \times \alpha_{m \to r}, & s_m = \text{copy}, \end{cases}$$

- $\delta(w_r^{(s)} = w^{(t)})$ is **indicator** function, taking the value 1 if the text of record $w_r^{(s)}$ is identical to the target word $w^{(t)}$
- Probability of **copying** record r from source is $\delta\left(s_m = \text{copy}\right) \times \alpha_{m \to r}$,which is the product
- of copy probability by local attention.
- The attention weights $\boldsymbol{\alpha}_m$ are computed from the **previous decoder state** $\boldsymbol{h}_{m-1}$
- The computation graph therefore remains a **feedforward network**, with recurrent paths such as

$$\boldsymbol{h}_{m-1}^{(t)} \to \boldsymbol{\alpha}_m \to w_m^{(t)} \to \boldsymbol{h}_m^{(t)}$$

Gen-Copy mechanism Alteration

- For end-to-end training, we switch the variable $\boxed{s_m}$ with a **gate** $\boxed{\pi_m}$
- This gate gets computed from a **two-layer feedforward network**, whose input consists of the **concatenation of the decoder state** $\boxed{\boldsymbol{h}_{m-1}^{(t)}}$

- The **attention-weighted representation** of the data $\boxed{\boldsymbol{c}_m = \sum_{r=1}^{R} \alpha_{m \to r} \boldsymbol{z}_r}$

$$\boxed{\pi_m = \sigma(\Theta^{(2)} f(\Theta^{(1)} [\boldsymbol{h}_{m-1}^{(t)}; \boldsymbol{c}_m]))}$$

- The full **generative probability** at token $\boxed{m}$ is then

$$\boxed{\mathrm{p}(w^{(t)} \mid \boldsymbol{w}_{1:m}^{(t)}, \mathbf{Z}) = \pi_m \times \underbrace{\frac{\exp \boldsymbol{\beta}_{w^{(t)}} \cdot \boldsymbol{h}_{m-1}^{(t)}}{\sum_{j=1}^{V} \exp \boldsymbol{\beta}_j \cdot \boldsymbol{h}_{m-1}^{(t)}}}_{\text{generate}} + (1 - \pi_m) \times \underbrace{\sum_{r=1}^{R} \delta(w_r^{(s)} = w^{(t)}) \times \alpha_{m \to r}}_{\text{copy}}}$$

Ryerson University

Text-to-text generation

- Text-to-text generation includes problems of **summarization** and **simplification**.

- These problems can be approached in two ways:
  - 1. Through **encoder-decoder architecture**
  - 2. By **operating** directly on the **input text**.

Text-to-text generation : summarization

- Sentence **summarization** is the task of shortening a sentence while preserving its **meaning**

- Sentence **summarization** is closely related to sentence compression, in which the summary is produced by **deleting** words or phrases from the original

  → Russian defense minister Ivanov called sunday for the creation of a joint front for combating global terrorism.

  → Russia calls for joint front against terrorism.

- Also, a **sentence** summary can also introduce **new words**.

Ryerson University

Neural abstractive summarization

- **Sentence summarization** can be treated as a **machine translation** problem, using the **attentional encoder-decoder** translation model
- The **longer sentence** is encoded into a sequence of vectors, one for each token.
- The **decoder** then **computes attention over these vectors** when updating its own recurrent state.
- As with data-to-text generation, it can be useful to boost the encoder-decoder model with the ability to **copy words directly from the source**

Long documents, fear of repetition

- **Problem :** When summarizing longer documents, an additional concern is that the summary not be repetitive, therefore, each part of the summary should cover new ground.

- **Solution :** This can be addressed by maintaining a vector of the sum total of all attention values thus far:

$$\boxed{\boldsymbol{t}_m = \sum_{n=1}^{m} \boldsymbol{\alpha}_n}$$

- This total can be used as an additional input to the computation of the attention weights, which enables the model to learn to prefer parts of the source which have not been attended to yet

$$\boxed{\alpha_{m\to n} \propto \exp\left(\boldsymbol{v}_\alpha \cdot \tanh(\Theta_\alpha[\boldsymbol{h}_m^{(t)}; \boldsymbol{h}_n^{(s)}; \boldsymbol{t}_m])\right)}$$

Ryerson University

Coverage Loss

- To encourage diversity in the generated summary, introduce a **coverage loss** to the objective function:

$$\ell_m = \sum_{n=1}^{M^{(s)}} \min(\alpha_{m\to n}, t_{m\to n})$$

- This loss will be low if α_m assigns little attention to words that already have large values in t_m
- Coverage loss is similar to the concept of marginal relevance, in which the reward for adding new content is proportional to the extent to which it increases the overall amount of information conveyed by the summary

Sentence fusion for multi-document summarization

- In multi-document summarization, the goal is to produce a summary that covers all the content of several documents
- One approach is to identify sentences across all documents that relate to a single theme, and then to fuse them into a single sentence
- Dependency parsing is often used as a technique for sentence fusion.

Sentence fusion for multi-document summarization

- After parsing each sentence, the resulting dependency trees can be aggregated into a lattice or or a graph structure.
- In this graph, identical or closely related words are fused into a single node
- The resulting graph can then be pruned back to a tree by solving an integer linear program

$$\max_{\boldsymbol{y}} \quad \sum_{i,j,r} \psi(i \xrightarrow{r} j, \boldsymbol{w}; \boldsymbol{\theta}) \times y_{i,j,r}$$

$$\text{s.t.} \quad \boldsymbol{y} \in \mathcal{C},$$

Ryerson University

$$\max_{\boldsymbol{y}} \quad \sum_{i,j,r} \psi(i \xrightarrow{r} j, \boldsymbol{w}; \boldsymbol{\theta}) \times y_{i,j,r}$$
$$\text{s.t.} \quad \boldsymbol{y} \in \mathcal{C},$$

- the variable $y_{i,j,r} \in \{0, 1\}$ indicates whether there is an edge from i to j of type r
- the score of this edge is $\psi(i \xrightarrow{r} j, \boldsymbol{w}; \boldsymbol{\theta})$, and C is a set of constraints, which ensures that y forms a valid dependency graph.
- As usual, w is the list of words in the graph, and θ is a vector of parameters.
- The score $\psi(i \xrightarrow{r} j, \boldsymbol{w}; \boldsymbol{\theta})$ reflects the "importance" of the modifier j to the overall meaning.

Sentence fusion for multi-document summarization

- **Linearization** is like the inverse of **dependency parsing**: instead of parsing from a sequence of tokens into a tree, we must convert the tree back into a sequence of tokens.
- This is typically done by generating a set of candidate linearizations, and choosing the one with the highest score under a language model

Ryerson University

Dialogue

- Dialogue systems are capable of **conversing** with a **human interlocutor**, often **to perform some task** , or **just to chat**
- Commercial systems such as **Alexa** and **Siri** have recently brought this technology into widespread use
- An example of Dialogue can be the following conversation to order pizza.

A: I want to order a pizza.
B: What toppings?
A: Anchovies.
B: Ok, what address?
A: The College of Computing building.
B: Please confirm: one pizza with artichokes, to be delivered to the College of Computing building.
A: No.
B: What toppings?

Ryerson University

Finite-state & agenda-based dialogue systems

- **Finite-state automata** is a formal model of computation, in which string inputs and outputs are linked to transitions between a **finite number of discrete states**.

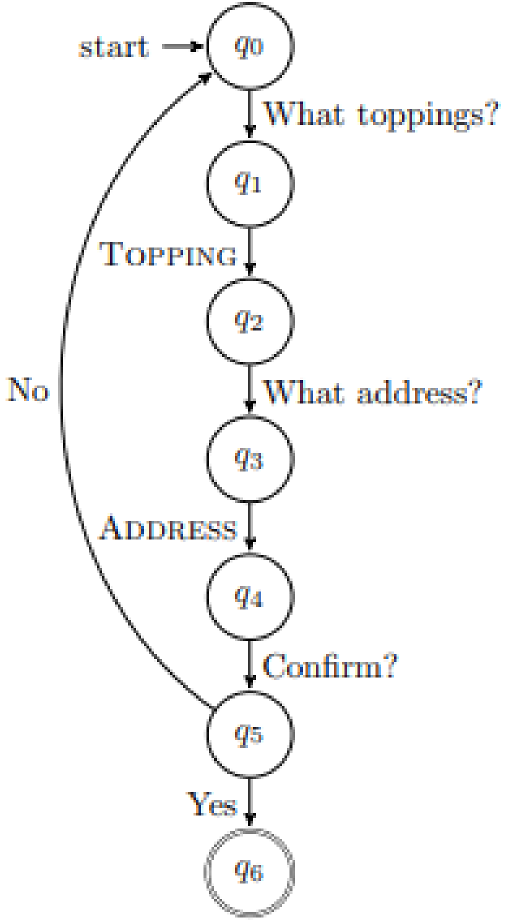
- This model fits **simple task-oriented dialogues**

- Dialogue can be represented with a **finite-state transducer**

- **Example** of **dialogue** and the associated **finite-state model**

- The TOPPING and ADDRESS are the two **slots** associated with the activity of ordering a pizza, which is called a **frame**.

A: I want to order a pizza.
B: What toppings?
A: Anchovies.
B: Ok, what address?
A: The College of Computing building.
B: Please confirm: one pizza with artichokes, to be delivered to the College of Computing building.
A: No.
B: What toppings?



simple task-oriented dialogue

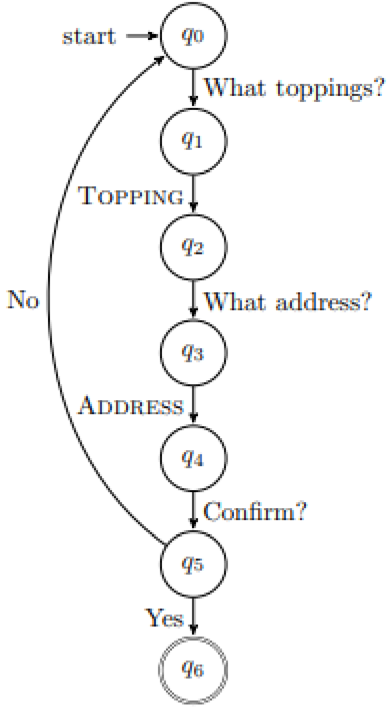
finite-state transducer

Finite-state & agenda-based dialogue systems

- **Problem:** In case the user wants to communicate more naturally using phrases like I'd, .. How should we approach?
- **Solution:** BIO-style sequence labeling. The tagger can be driven by a bi-directional RNN , similar to recurrent approaches to semantic role labeling

I'd like anchovies
O O B-TOPPING

A: I want to order a pizza.
B: What toppings?
A: Anchovies.
B: Ok, what address?
A: The College of Computing building.
B: Please confirm: one pizza with artichokes, to be delivered to the College of Computing building.
A: No.
B: What toppings?



simple task-oriented dialogue

finite-state transducer

Ryerson University

Dialogue : Neural chatbots

- **Chatbots** are systems that parry the user's input with a response that keeps the conversation going

- **Chatbots** can be built from the **Encoder-Decoder architecture**

- **Encoder** converts the user's input into a **vector**
- **Decoder** produces a sequence of words as a **response**

Dialogue : Neural chatbots

- **Problem :** Encoder-decoder models struggle to maintain coherence over **longer conversations**.

- **Solution :** Modeling the dialogue **context** recurrently.

- By creating **hierarchical recurrent network**, including both **word-level** and **turn-level** recurrences.

- The **turn-level hidden state** is then used as **additional context** in the **decoder**

Ryerson University

Dialogue : Neural chatbots

- The **encoder-decoder architecture** can be integrated into task-oriented dialogue systems.
- Neural chatbots can be trained **end-to-end:**
  - The **user's turn** is analyzed by the **encoder**
  - The **system output** is generated by the **decoder**.
- This architecture can be trained by **log-likelihood** using **backpropagation** or by more elaborate objectives, using **reinforcement learning**

Ryerson University

Task-oriented dialogue : MDP and RNN

- The **Task-oriented** dialogue systems, typically involve set of modules: one for recognizing the **user input**, another for deciding what **action to take**, and a third for arranging the text of the **system output**

- RNN decoders can be integrated into **Markov Decision Process** dialogue systems, by conditioning the decoder on a representation of the information that is to be expressed in each turn

Ryerson University

Task-oriented dialogue : Element Embedding

- Another promising direction is to **create embeddings** for the **elements** in the domain:
  - for example, the **slots** in a record and the **entities** that can fill them.
- The **encoder** then encodes not only the **words** of the **user's input**, but the **embeddings of the elements** that the user mentions.
- The **decoder** is able with the ability to **refer to specific elements** in the knowledge base.

Ryerson University

Thank you very much for listening

Ryerson University