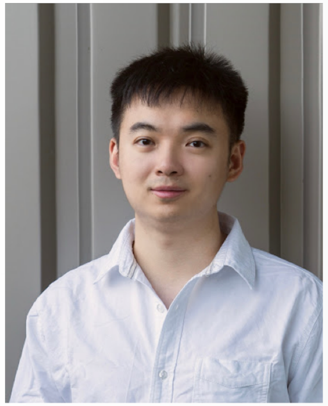
EMNLP 2021 Tutorial

Knowledge-Enriched Natural Language Generation

Wenhao Yu[1], Meng Jiang[1], Zhiting Hu[2], Qingyun Wang[3], Heng Ji[3], Nazneen Rajani[4]

1 University of Notre Dame 2 University of California San Diego
3 University of Illinois at Urbana-Champaign 4 Salesforce Research

General Methods of Knowledge + NLG

This part: General principles and methodologies for integrating knowledge into NLG

Next part (by Wenhao): Concrete examples and instantiations of the general methods in recent NLG works

General Methods of Knowledge + NLG

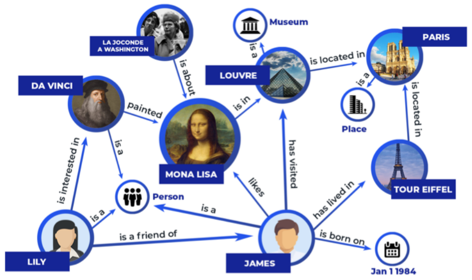
This part: General principles and methodologies for integrating knowledge into NLG

Overview:

- Knowledge-enhanced model architectures
  - Attention/copy mechanisms
  - Graph neural models
- Knowledge-enhanced learning
  - Auxiliary **loss/tasks**
  - Reinforcement learning with knowledge-informed **rewards**
  - Learning with knowledge **constraints**
- Knowledge-enhanced inference
  - Steered decoding
  - Prompts

Knowledge-enhanced model architectures

Bake knowledge into the model through specific architectures



Graph NNs for modeling graph-structured knowledge (e.g., knowledge graphs)

K Knowledge

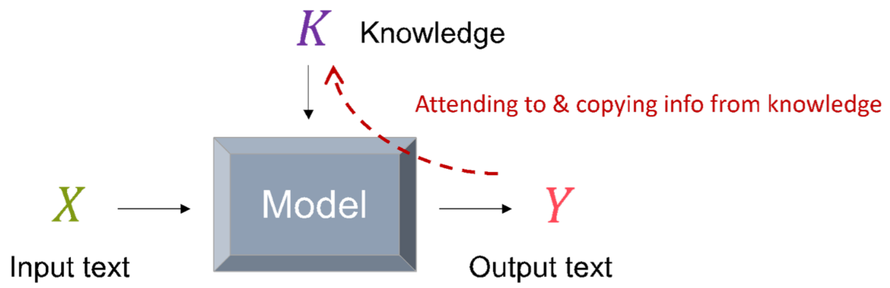
Attending to & copying from knowledge

X → Model → Y

Input text

Output text

Architectures (I): Attention Mechanisms



- Chooses which information to pay attention to

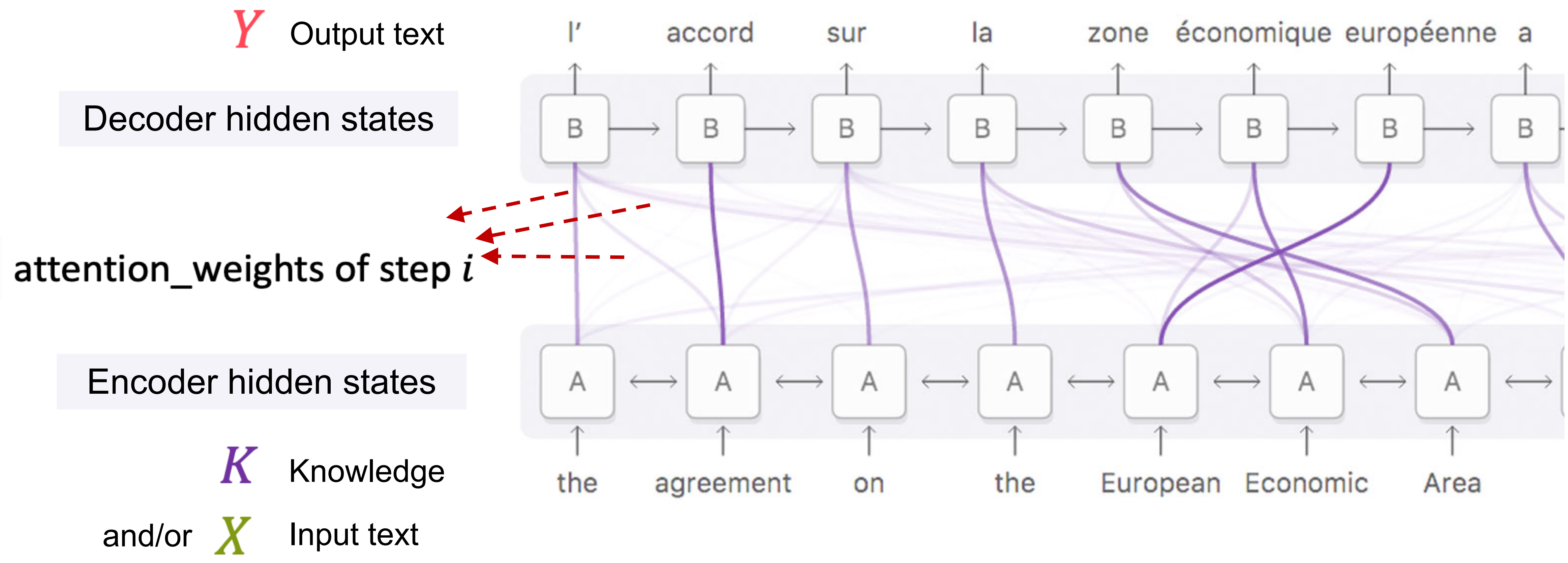
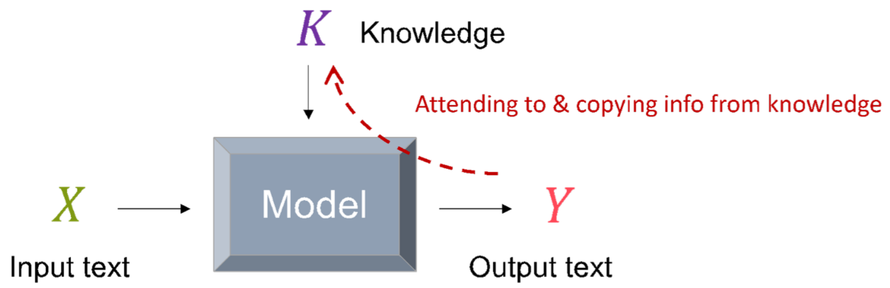


Figure courtesy: Olah & Carter,

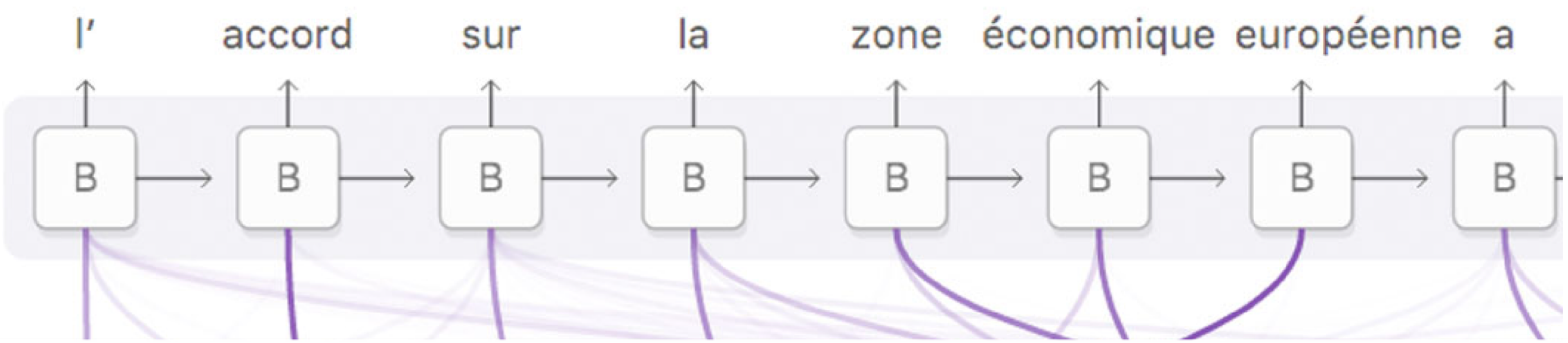
Architectures (I): Attention Mechanisms



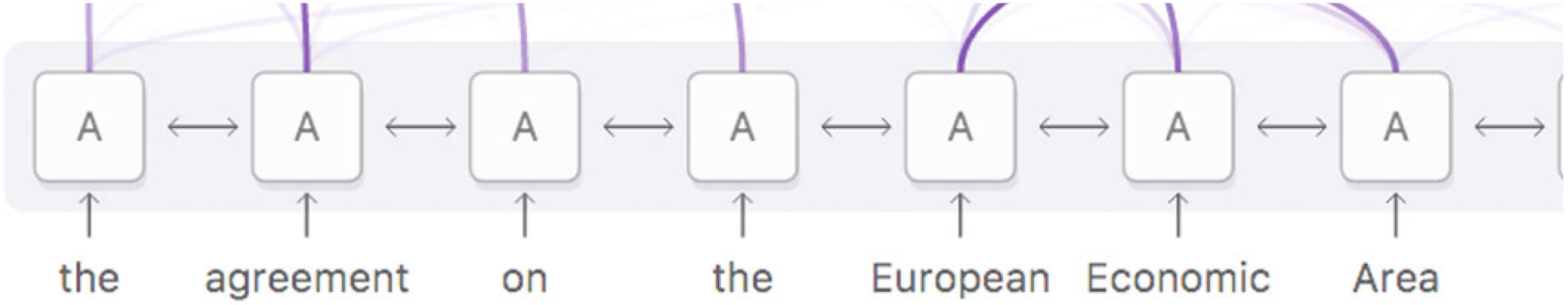
- Chooses which information to pay attention to

Y Output text

Decoder hidden states



attention_weights of step i = softmax(alignment_scores(decoder_state i, encoder_states))



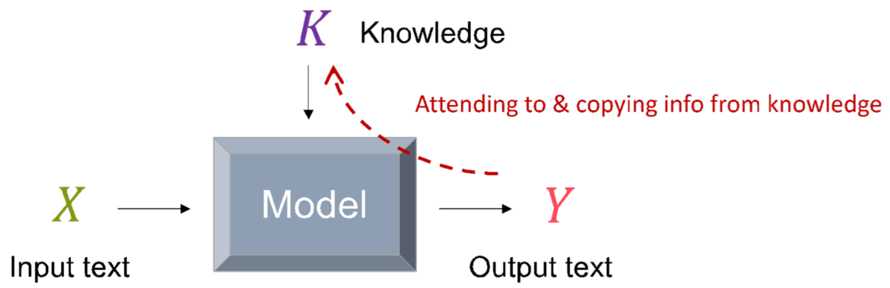
Encoder hidden states

K Knowledge

and/or X Input text

Figure courtesy: Olah & Carter,

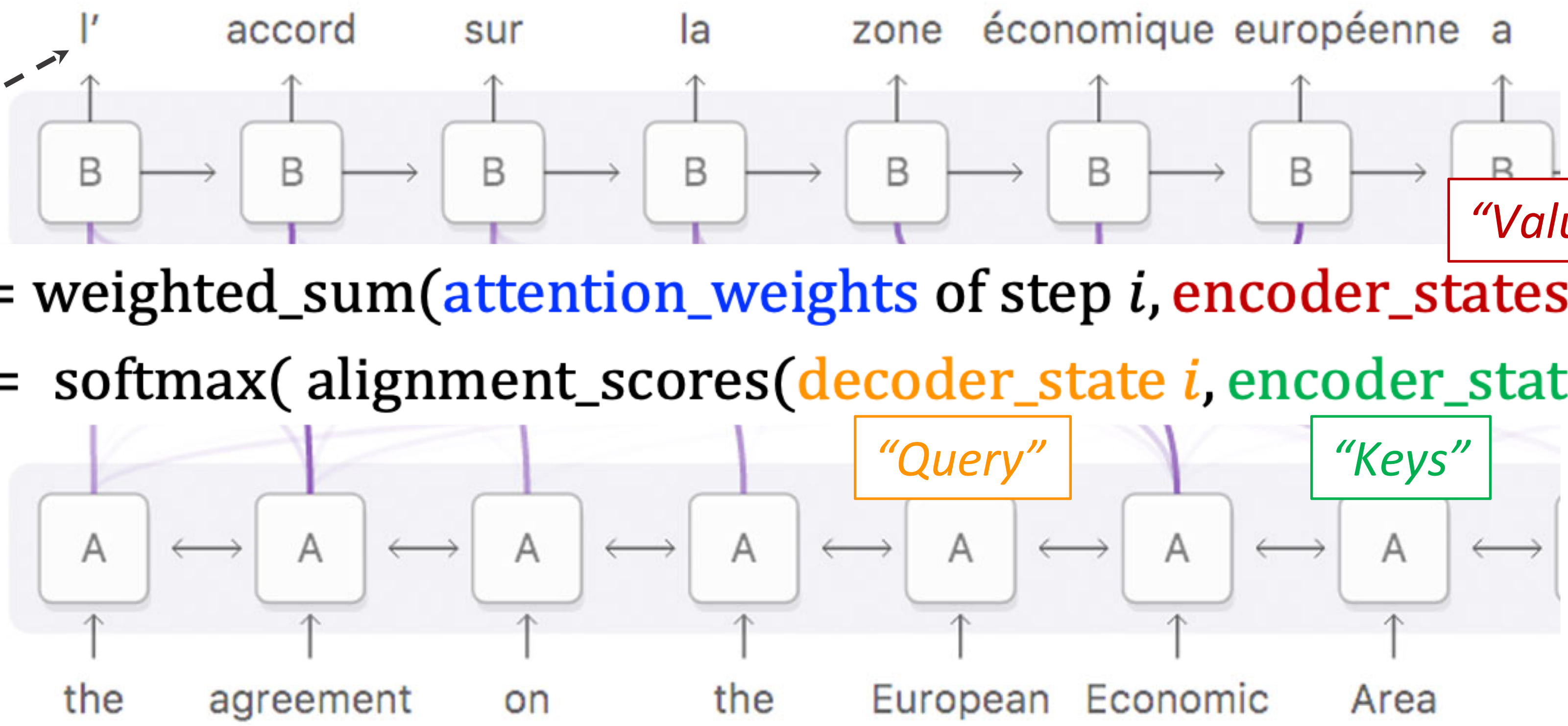
Architectures (I): Attention Mechanisms



- Chooses which information to pay attention to

Y Output text

Generate output token i :es



context vector of step i = weighted_sum(attention_weights of step i, encoder_states)

attention_weights of step i = softmax(alignment_scores(decoder_state i, encoder_states))

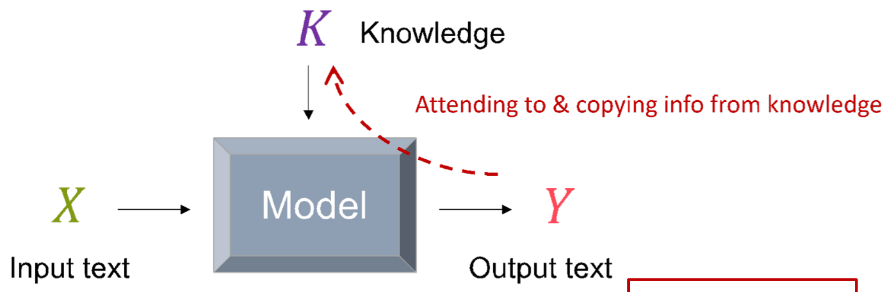
Encoder hidden states

K Knowledge

and/or X Input text

Figure courtesy: Olah & Carter,

Architectures (I): Attention Mechanisms



•

"Values"

context vector of step i = weighted_sum(attention_weights of step i, encoder_states)

attention_weights of step i = softmax(alignment_scores(decoder_state i, encoder_states))

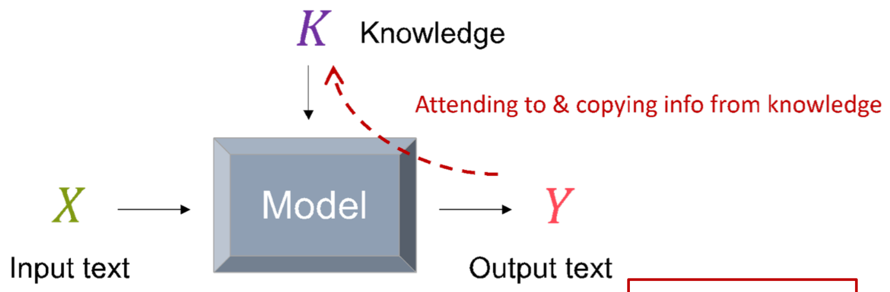
"Query" "Keys"

- Variations of attention mechanisms
  - Different alignment_scores functions

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

Figure courtesy: Lilian Weng

Architectures (I): Attention Mechanisms



•

“Values”

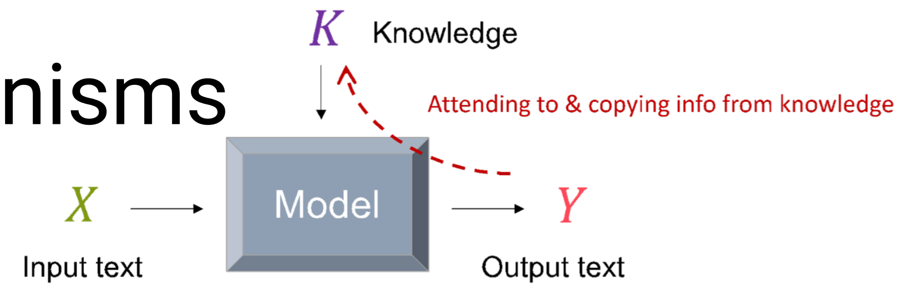
context vector of step i = weighted_sum(attention_weights of step i, encoder_states)

attention_weights of step i = softmax(alignment_scores(decoder_state i, encoder_states))

“Query” “Keys”

- Variations of attention mechanisms
  - Different alignment_scores functions
  - Self attention: Query = Keys = Values
  - Multi-head attention (Transformers)
  - Kernelized attention
  - …

Architectures (II): Copy/Pointing Mechanisms



- Copy relevant information to the output text

Probability of choosing the **generation mode**

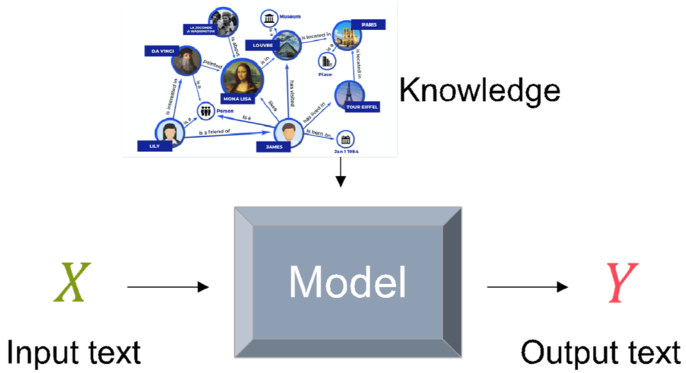
Probability of choosing the **copy mode**

$$p(y_t) = p_m \cdot p_{gen}(y_t) + (1 - p_m) \cdot p_{copy}(y_t)$$

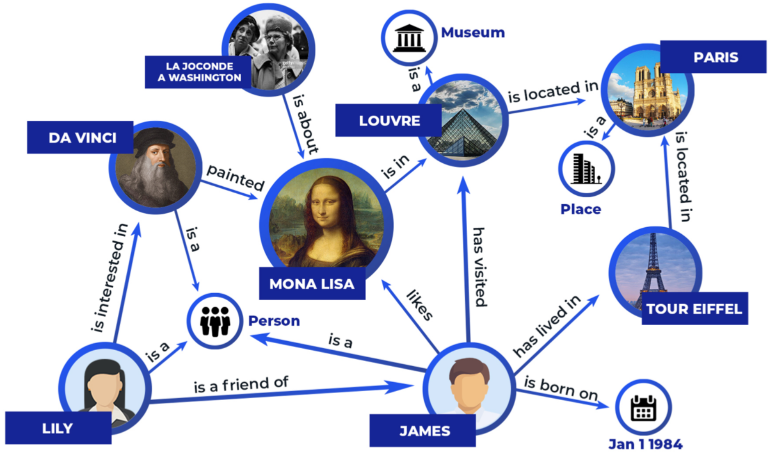
Probability of generating the token y_t

Probability of copying the token y_t from knowledge / input

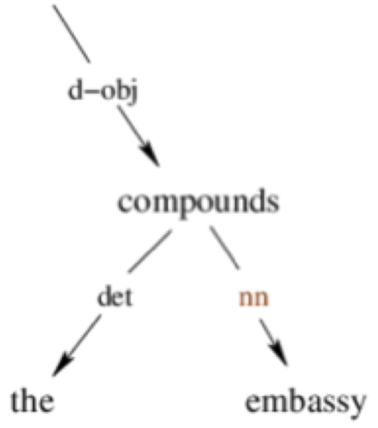
Architectures (III): Graph Neural Models



- Representation and reasoning over graph-structured knowledge
- Bridge the gap between graph representation and text generation

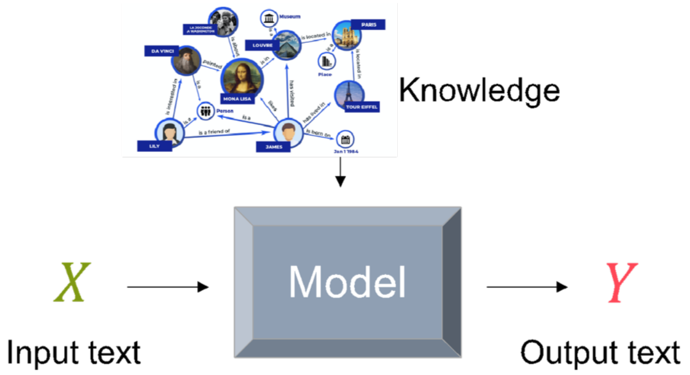


Knowledge graphs (KGs)



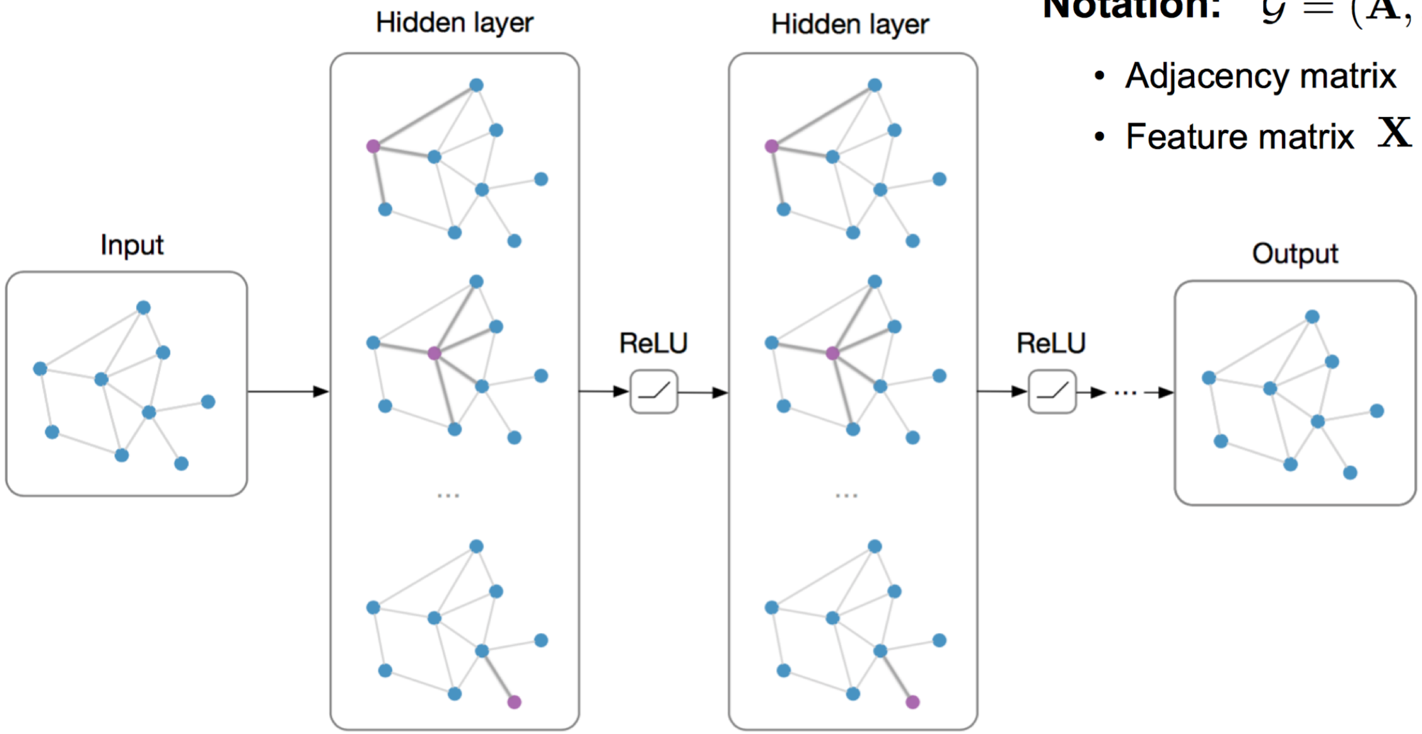
Dependency graphs

Architectures (III): Graph Neural Models



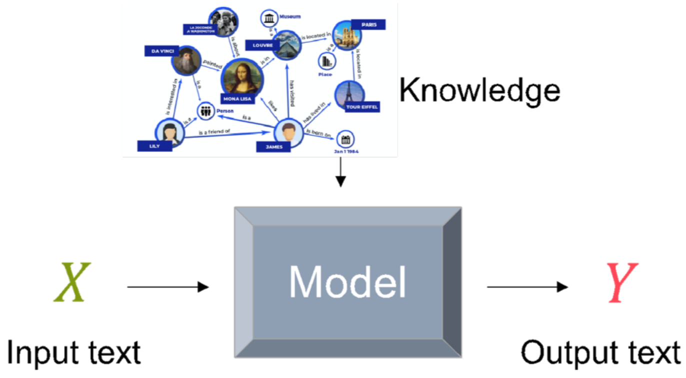
Notation: $\mathcal{G} = (\mathbf{A}, \mathbf{X})$

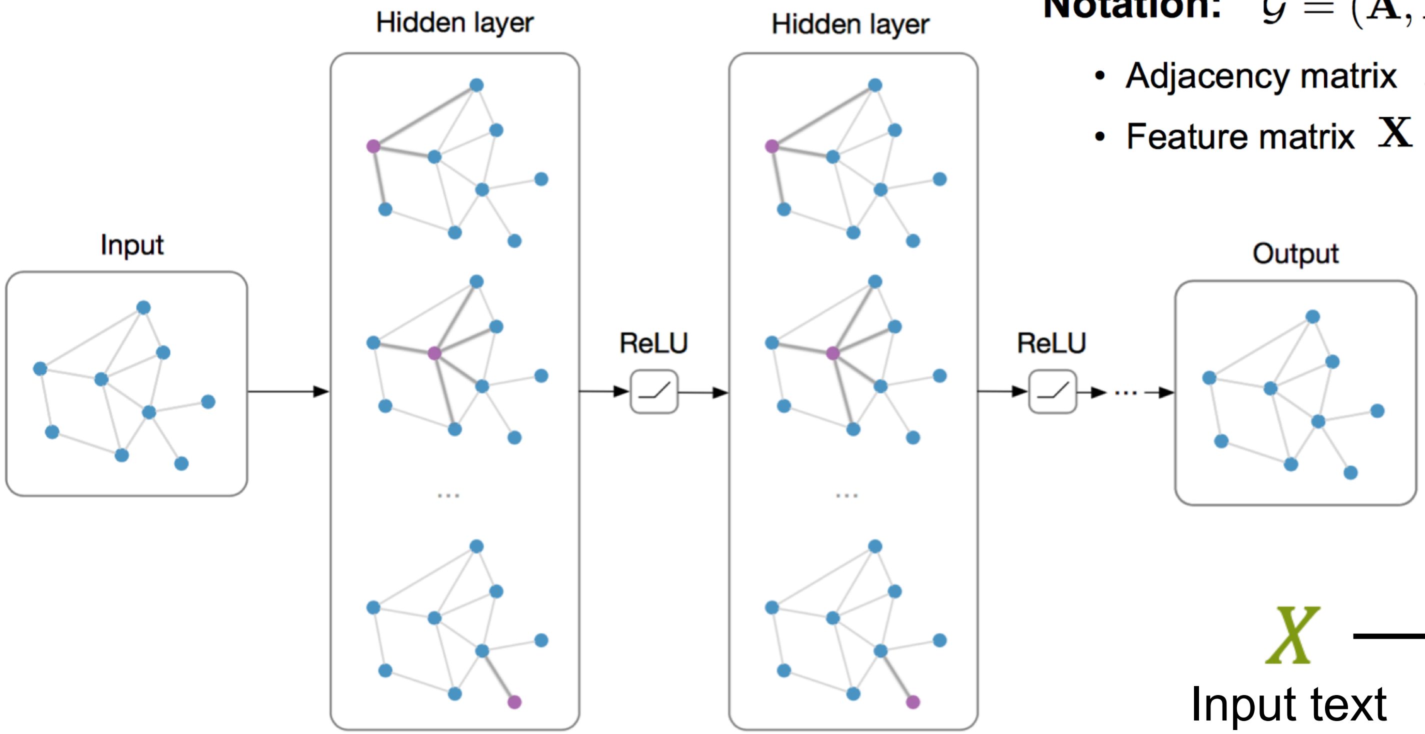
- Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$
- Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$



Main idea: Pass massages between nodes to refine node (and possibly edge) representations

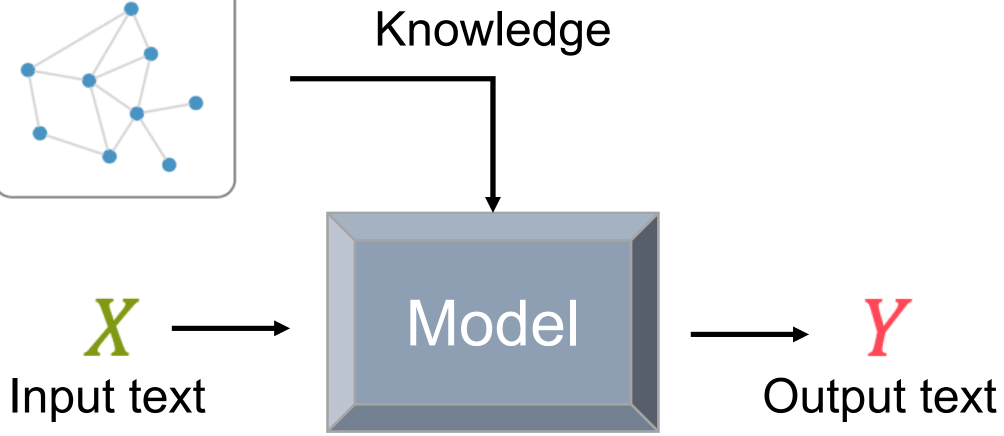
Architectures (III): Graph Neural Models





Notation: $\mathcal{G} = (\mathbf{A}, \mathbf{X})$

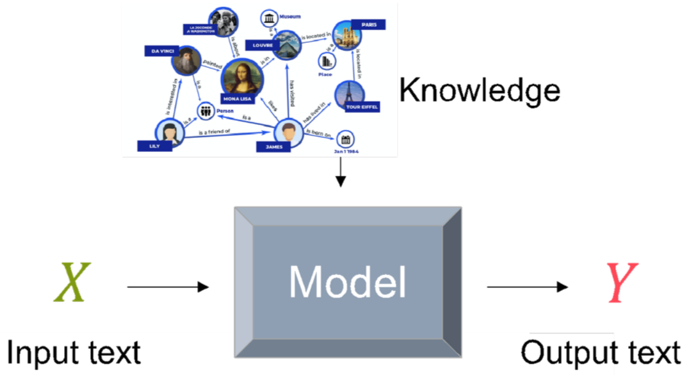
- Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$
- Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$



Main idea: Pass massages between nodes to refine node (and possibly edge) representations

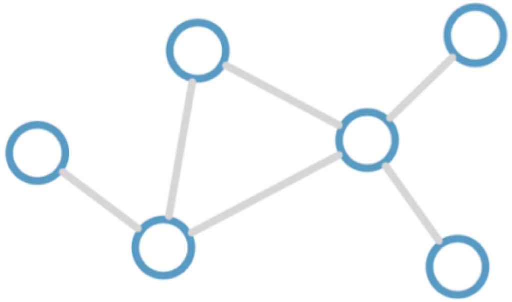
Slide courtesy: Thomas Kipf

Architectures (III): Graph Neural Models

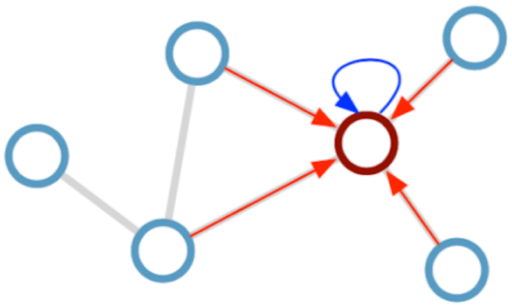


Graph Convolutional Networks (GCNs), Kipf & Welling 2017

Consider this undirected graph:



Calculate update for node in red:



Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

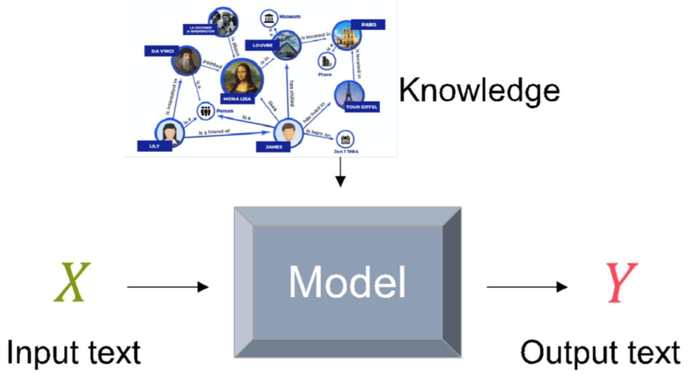
Scalability: subsample messages [Hamilton et al., NIPS 2017]

$\mathcal{N}_i$: neighbor indices

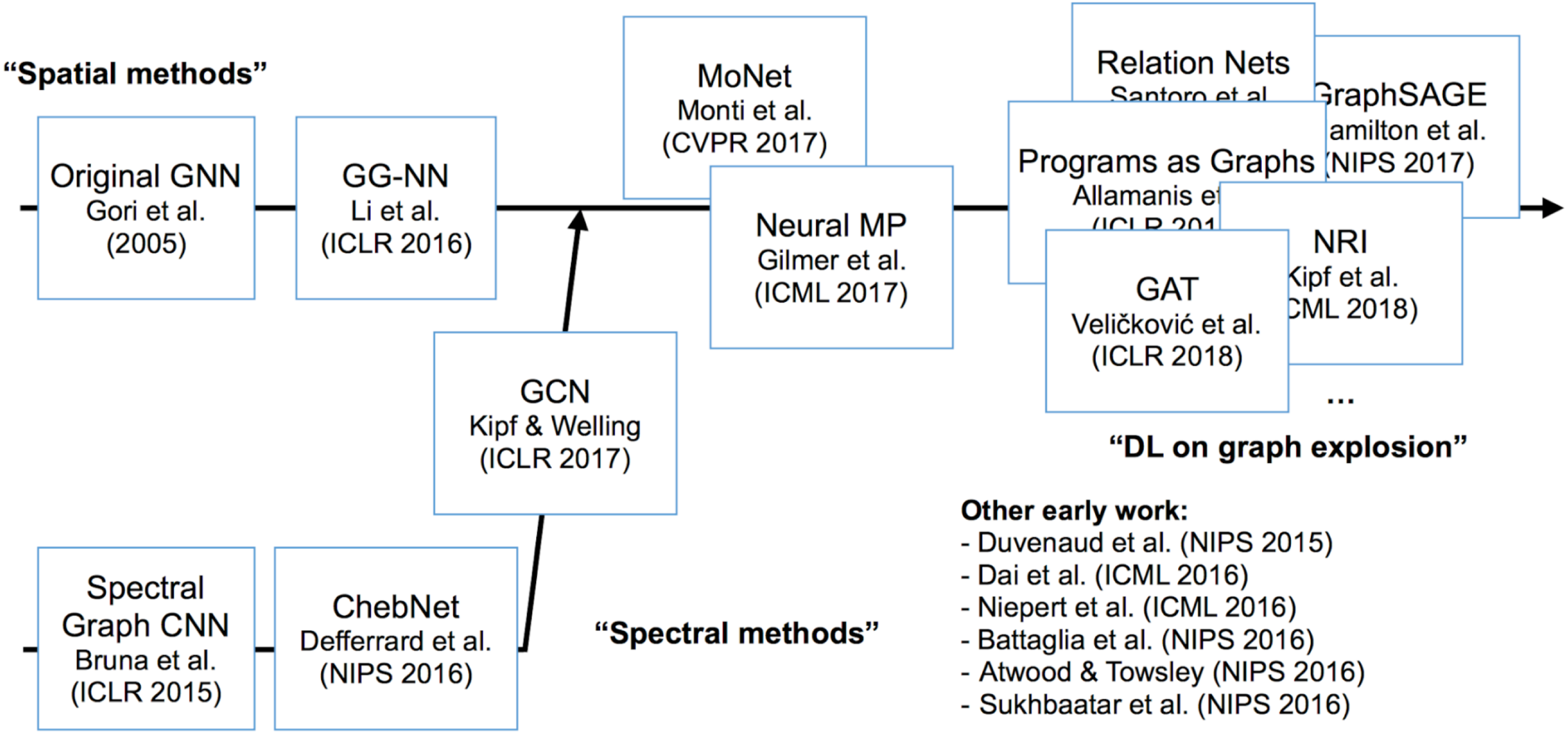
c_{ij} : norm. constant (fixed/trainable)

Slide courtesy: Thomas Kipf

Architectures (III): Graph Neural Models



A brief history of graph neural networks



Other early work:
- Duvenaud et al. (NIPS 2015)
- Dai et al. (ICML 2016)
- Niepert et al. (ICML 2016)
- Battaglia et al. (NIPS 2016)
- Atwood & Towsley (NIPS 2016)
- Sukhbaatar et al. (NIPS 2016)

(slide inspired by Alexander Gaunt's talk on GNNs)

Slide courtesy: Thomas Kipf

General Methods of Knowledge + NLG

This part: General principles and methodologies for integrating knowledge into NLG

Overview:

- Knowledge-enhanced model architectures
  - Attention/copy mechanisms
  - Graph neural models
- Knowledge-enhanced learning
  - Auxiliary **loss/tasks**
  - Reinforcement learning with knowledge-informed **rewards**
  - Learning with knowledge **constraints**
- Knowledge-enhanced inference
  - Steered decoding
  - Prompts

Knowledge-enhanced learning

- Design knowledge-informed learning problems
  - Auxiliary tasks
  - Reward
  - Constraints
- Model is trained to solve the problems
  - So that knowledge information is absorbed into model parameters
- Often agnostic to model architectures:
  - Thus, can combine the learning methods with any knowledge-enhanced architectures we’ve just seen

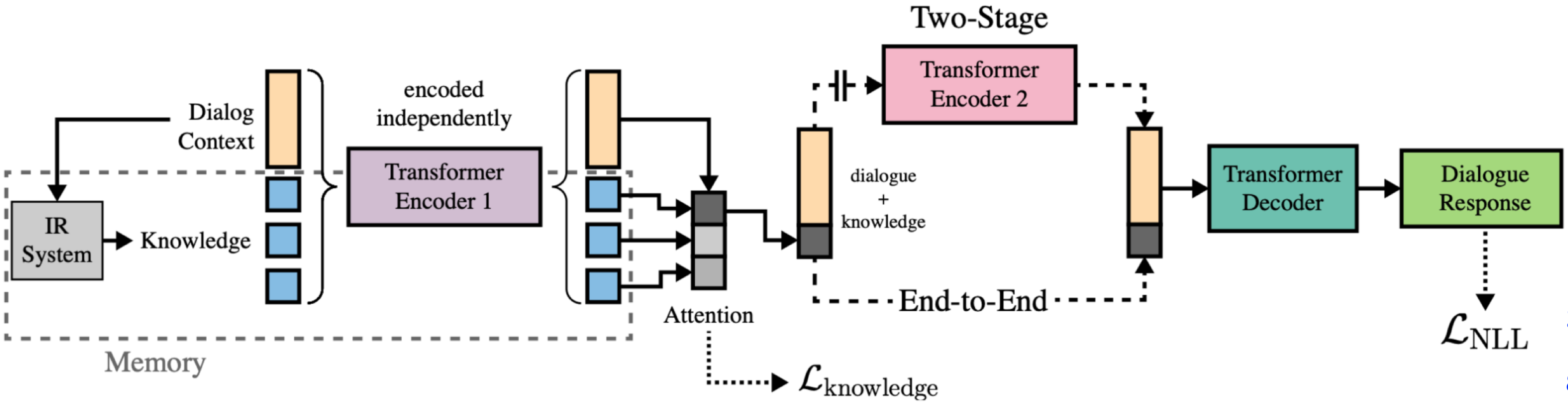
Learning (I): Auxiliary tasks

- “Knowledge as target”
  - Create learning targets (labels) based on the knowledge
  - Use the targets to supervise the training of the model

Learning (I): Auxiliary tasks

(1) Combine the auxiliary tasks with standard text generation task

- Lead to a *multi-task* learning paradigm
- Ex: dialog generation



standard dialog generation loss

auxiliary loss to reconstruct sentences from Wikipedia

Dinan et al., "Wizard of Wikipedia: Knowledge-Powered Conversational agents"

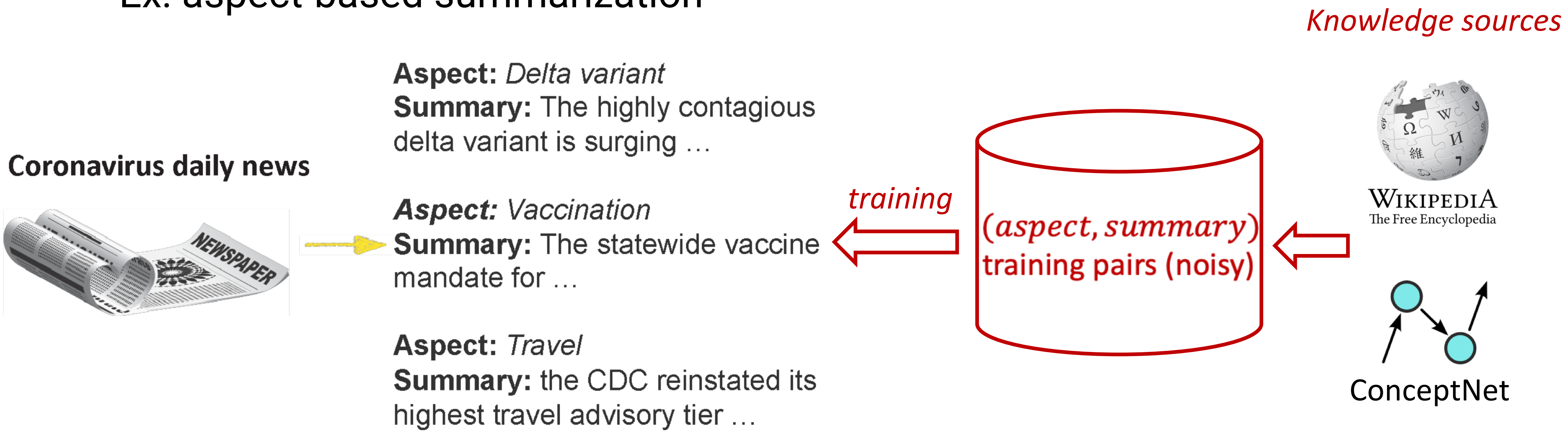
Learning (I): Auxiliary tasks

(1) Combine the auxiliary tasks with standard text generation task

- Lead to a *multi-task* learning paradigm

(2) The auxiliary tasks provide direct supervision for the text generation task

- Lead to a *weakly-supervised* learning paradigm
- Ex: aspect-based summarization

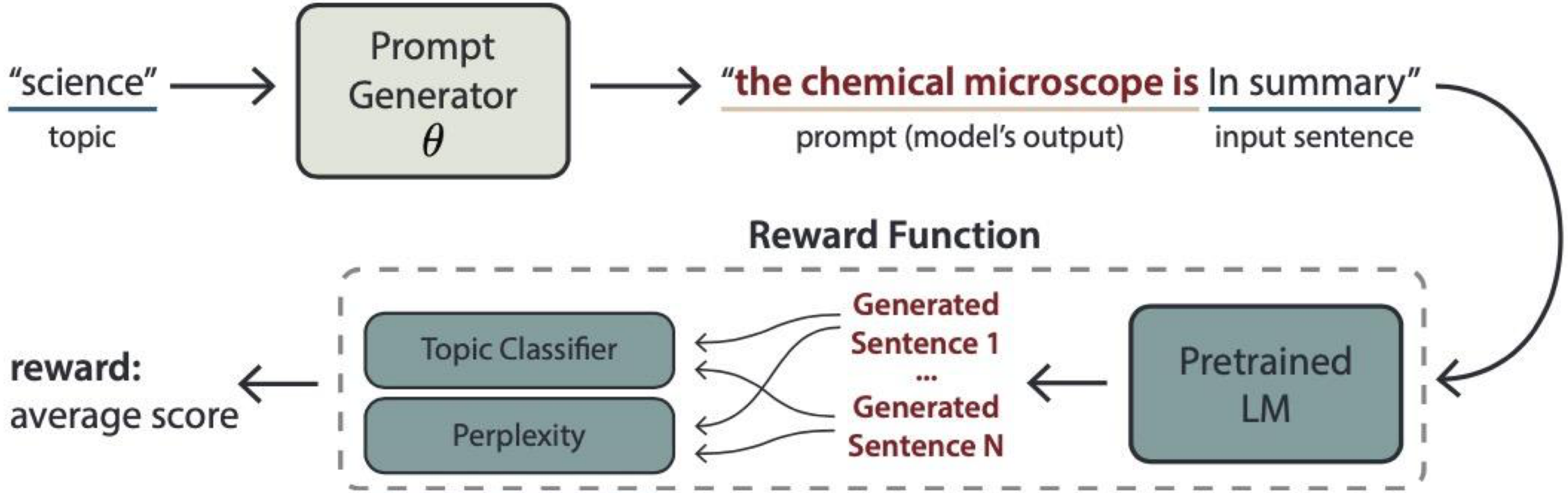


Tan et al., "Summarizing Text on Any Aspects: A Knowledge-Informed Weakly-Supervised

Learning (II): Reinforcement learning

- "Knowledge as reward"
  - Knowledge-informed reward function evaluates the quality of generation
  - Model is trained to maximize the reward using reinforcement learning:
    - Policy gradient, (Soft) Q-learning, etc.

- Ex: Learning to generate prompts for topic-controllable generation



Han et al., "Text Generation with Efficient (Soft) Q-Learning"

Learning (III): Learning with knowledge constraints

- "Knowledge as constraints"
  - Impose knowledge-informed constraints on the NLG training objective
  - Model is trained to optimize the objective subject to the constraints
- Methods: posterior regularization, constraint-driven learning, integer linear programming, …
  - Posterior regularization:

Minimize KL divergence: encourage model p_θ to stay close to auxiliary distribution q

Standard NLG objective

$$\min_{\theta, q, \boldsymbol{\xi} \geq 0} \mathcal{L}(\boldsymbol{\theta}) + \mathrm{KL}(q(\boldsymbol{y}|\boldsymbol{x}) \,||\, p_\theta(\boldsymbol{y}|\boldsymbol{x})) + ||\boldsymbol{\xi}||_b$$

$$s.t.\ \mathbb{E}_q\left[f(\boldsymbol{x}, \boldsymbol{y})\right] \leq \boldsymbol{\xi}$$

Impose constraints on q

Solve with an EM-style procedure

Learning (III): Learning with knowledge constraints

- "Knowledge as constraints"
  - Impose knowledge-informed constraints on the NLG training objective
  - Model is trained to optimize the objective subject to the constraints
- Methods: posterior regularization, constraint-driven learning, integer linear programming, …
  - Posterior regularization
  - Ex:

General Methods of Knowledge + NLG

This part: General principles and methodologies for integrating knowledge into NLG

Overview:

- Knowledge-enhanced model architectures
  - Attention/copy mechanisms
  - Graph neural models
- Knowledge-enhanced learning
  - Auxiliary **loss/tasks**
  - Reinforcement learning with knowledge-informed **rewards**
  - Learning with knowledge **constraints**
- Knowledge-enhanced inference
  - Steered decoding
  - Prompts

Knowledge-enhanced inference

- Integrate knowledge during the text decoding process
- Can be applied to pretrained language models (e.g., GPT-2/3, T5) for knowledge-enhanced NLG

Inference (I): Steered decoding

- Guide the decoding by changing the generation distribution

- TODO: PPLM, GeDi, DeLorean

Inference (II): Prompts

- Guide the decoding by changing the generation distribution