Retrieval Augmented Generation: the idea of dual memory representations and Diagnostic Captioning

Georgios M. Moschovis

MSc Machine Learning – KTH Royal Institute of Technology

School of Electrical Engineering and Computer Science

geomos@kth.se

Retrieval Augmented Generation architecture

Adaptive Moments (Adam) optimizer uses exponentially decaying average to discard history and momentum as an estimate of the first-order gradient. It has bias corrections for first-order and second-order moments and converges rapidly after finding a local convex bowl.

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \varepsilon \frac{\mathbf{v}^{(t)}}{\delta + \sqrt{\mathbf{r}^{(t)}}},\ \delta,\ \varepsilon \in \mathbb{R}^+$$

$$\text{w.r.t. } \mathbf{v}^{(t+1)} = \rho_1 \mathbf{v}^{(t)} + (1-\rho_1)\mathbf{g}^{(t)},\ \mathbf{r}^{(t+1)} = \rho_2 \mathbf{r}^{(t)} + (1-\rho_2)\left(\mathbf{g}^{(t)}\right)^2,\ \rho_1,\ \rho_2 \in \mathbb{R}^+$$

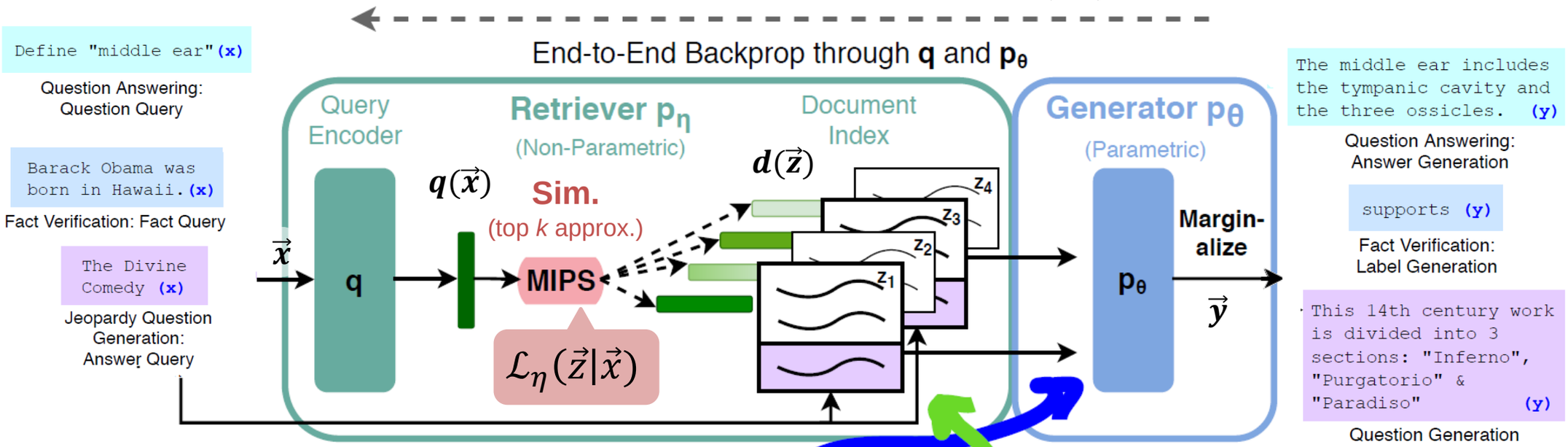


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder + Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

[Extracted from 2005.11401; slightly modified for explanation purposes.]

Data Representation with dual-memory

- Dual memory components are pre-trained and pre-loaded with extensive knowledge to encapsulate information via the representations without further training:
  - the **generator p_θ** acts as a parametric memory
  - the **retriever p_η** embodies a non-parametric memory in the query encoder **q** that can be modeled as transformer network (e.g. BERT) to create suitable representations.
- To train the retriever p_η and generator p_θ end-to-end we can treat the retrieved document as a latent variable $\boldsymbol{z}$ while the embedding of the closest doc. representation is represented as $\mathbf{d}(\boldsymbol{z})$). **Maximum Inner Product Search (MIPS)** algorithm is used to compute the top *k* retrieved documents $\left\{\boldsymbol{z}_i\right\}_{i=1}^{k}$.
- We predict $\boldsymbol{y}$ by marginalizing over the predictions.

Dense Passage Retriever (DPR)

- Assume that our collection $\mathcal{D}$ contains $|\mathcal{D}| = D$ documents, thus $\mathcal{D} = \left\{\boldsymbol{z_i}\right\}_{i=1}^{D}$ that are split into passages of equal lengths and correspond to basic retrieval units.
  - The corpus takes the form $\mathcal{C} = \left\{p_i\right\}_{i=1}^{k}$.
  - Each passage $p \in \mathcal{C}$ can be interpreted as a sequence of tokens $p_i = \left\{w_j^{(i)}\right\}_{i=1}^{|p_i|}$.
- The goal is to find a span $\left\{w_j^{(i)}\right\}_i$ from one of the passages that answers the question. The idea intuitively associates each of the questions to a filtered subset of the training corpus with passages $p_i \in \mathcal{C}$ that ideally answer the question.
  - The corresponding function is R: $(q, \mathcal{C}) \longrightarrow \mathcal{C}_{\mathcal{f}}$.
  - The amount of the retrieved passages is typically smaller than the corpus size $\left|\mathcal{C}_{\mathcal{f}}\right| << |\mathcal{C}|$.

Dense Passage Retriever (DPR)

- The Dense Passage Retriever implementation incorporates the use of FAISS (Fast AI Similarity Search) from Facebook. It provides several advantages:
  - GPU optimization due to improved parallelism
  - improved preprocessing of the doc. collection $\mathcal{D} = \left\{ \boldsymbol{z_i} \right\}_{i=1}^{D}$
  - uses Inverted File Index to also cope with clustering
  - is trainable: captures better semantic similarities
  - the final (optional) step is data quantization
- Of course there are several disadvantages as well:
  - lower performance with out-of-vocabulary words
  - training the model takes time: slower compared to TF-IDF
  - it is comprised of two phases: indexing and retrieval
    → computationally expensive (reader-retriever)

Dense Passage Retriever (DPR)

- The **retriever p_η** provides latent documents conditioned on the input. In other words, it codifies a text document $\boldsymbol{x} \in \mathcal{D}$ into dense numerical representations.
- The parameterized retrieval component as an overall block produces $p_\eta(\boldsymbol{z}|\boldsymbol{x})$ and follows the encoder-decoder structure based on DPR. The likelihood of the latent factors $\boldsymbol{z}$ that are associated to the images $\boldsymbol{x}$ is proportional to:

$$p_\eta(\boldsymbol{z} \mid \boldsymbol{x}) \propto \exp\left(\mathbf{d}(\boldsymbol{z})^T \mathbf{q}(\boldsymbol{x})\right),$$

$$\text{where } \mathbf{d}(\boldsymbol{z}) = \text{BERT}_d(\boldsymbol{z}),\ \mathbf{q}(\boldsymbol{x}) = \text{BERT}_q(\boldsymbol{x})$$

- The **Maximum Inner Product Search (MIPS)** algorithm then returns the latent representations of the text documents with the highest probability.

Dense Passage Retriever (DPR)

- The **retriever p_η** provides latent documents conditioned on the in
- The p
prod
based
associated to the images it is proportional to:

The encoder is a *Siamese Network*

This type of networks that compute probabilities using the inner product below are also referred to in ANNs bibliography as *Siamese* or *bi-encoder* or *two towers networks*.

$$p_\eta(\boldsymbol{z} \mid \boldsymbol{x}) \propto \exp\left(\mathbf{d}(\boldsymbol{z})^T \mathbf{q}(\boldsymbol{x})\right)$$

$$p_\eta(\boldsymbol{z} \mid \boldsymbol{x}) \propto \exp\left(\mathbf{d}(\boldsymbol{z})^T \mathbf{q}(\boldsymbol{x})\right),$$

$$\text{where } \mathbf{d}(\boldsymbol{z}) = \text{BERT}_d(\boldsymbol{z}),\ \mathbf{q}(\boldsymbol{x}) = \text{BERT}_q(\boldsymbol{x})$$

- The **Maximum Inner Product Search (MIPS)** algorithm then returns the latent representations of the image captions with the highest probability.

Dense Passage Retriever (DPR)

- The **retriever p_η** provides latent documents conditioned on the in D into c
- The p prod e based e associated to the image in the pre-trained set.

Document and query encoder

The document encoder $\mathbf{d}(\boldsymbol{z})$ is trained once, during pre-train phase, whereas the query encoder $\mathbf{q}(\boldsymbol{x})$ is trained continuously end-to-end by applying back-propagation with Adam optimizer. This way, query encoder learns how to adjust the weights to retrieve better for a downstream task.

$$p_\eta(\boldsymbol{z} \mid \boldsymbol{x}) \propto \exp\left(\mathbf{d}(\boldsymbol{z})^T \mathbf{q}(\boldsymbol{x})\right),$$

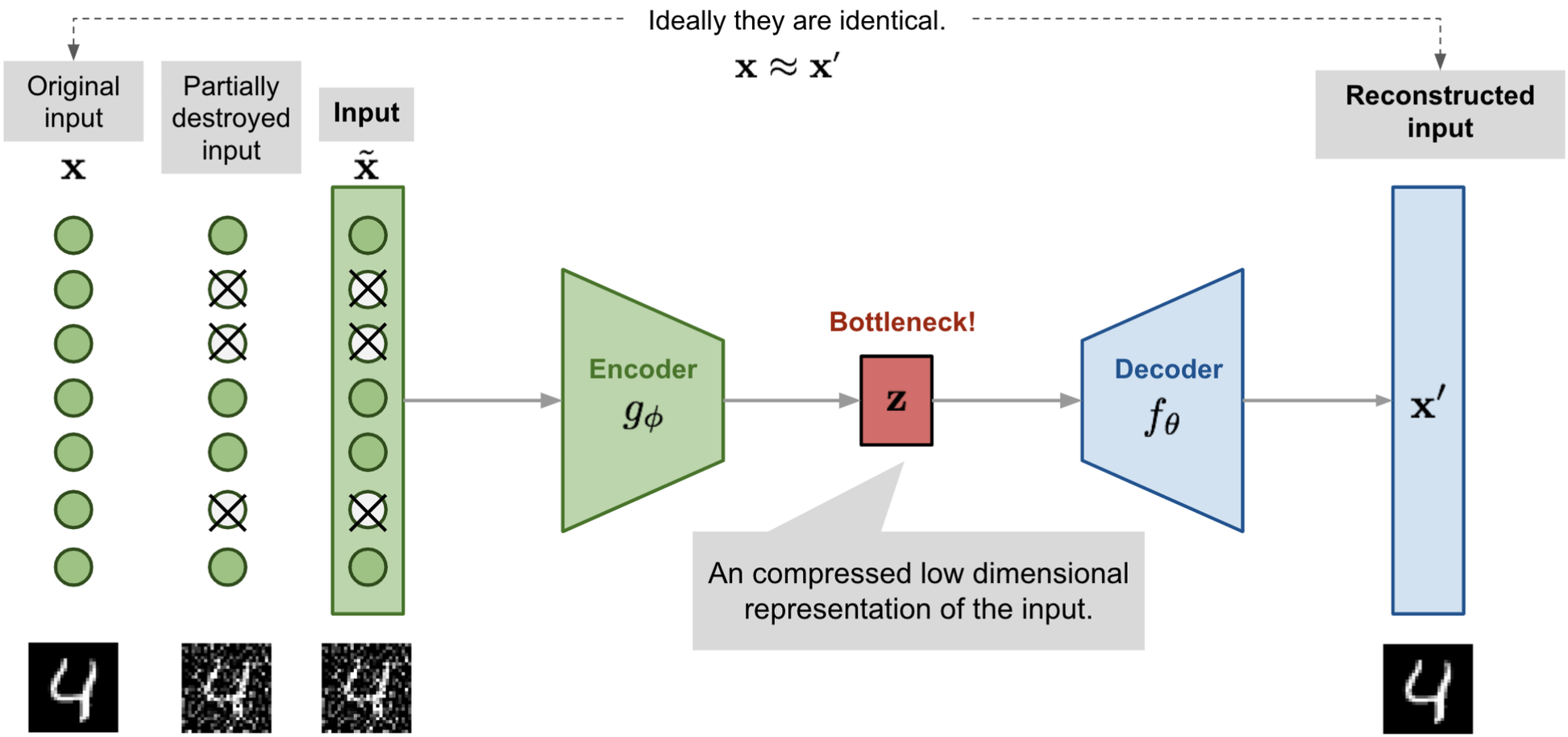
$$\text{where } \mathbf{d}(\boldsymbol{z}) = \text{BERT}_d(\boldsymbol{z}),\ \mathbf{q}(\boldsymbol{x}) = \text{BERT}_q(\boldsymbol{x})$$

- The **Maximum Inner Product Search (MIPS)** algorithm then returns the latent representations of the image captions with the highest probability.

Generator: seq2sec model

- The **generator p_θ** is a seq2seq model, most probably instance of a transformer network with a generative part as BART, T5, GPT2, which conditions on the latent documents $\boldsymbol{z}$ together with the input $\boldsymbol{x}$ to generate the output.
  - Since **BART** (denoising autoencoder) is the current state-of-the-art in text generation we can use its parameters θ as our parametric memory.
  - We use latent documents $\boldsymbol{z}$ concatenated with query $\boldsymbol{x}$ in order to generate caption $\boldsymbol{y}$.
- As an overall component, it produces $p_\theta(y_i|x,z,y_{1:i-1})$ to create a Language Model (LM) over the tokens vocabulary $\mathcal{V}$ given in documents $\boldsymbol{z}$ and queries $\boldsymbol{x}$, which can be modeled using the transformer architecture.

Denoising Autoencoders and BART



1. A training sample is sampled from the training data.
2. A corrupted version of the sample $\boldsymbol{x}$ is drawn from some noise model $\mathcal{M}_d\left(\tilde{\boldsymbol{x}} \middle| \boldsymbol{x}\right)$.
3. Pairs are used as a training sample to estimate the reconstruction distribution:

$$p_{\text{reconstruction}}\left(\boldsymbol{x} \middle| \tilde{\boldsymbol{x}}\right) = p_{\text{decoder}}\left(\boldsymbol{x} \middle| \boldsymbol{h}\right) = p_{\text{decoder}}\left(\boldsymbol{x} \middle| \boldsymbol{e}(\tilde{\boldsymbol{x}})\right)$$

[Fig. extracted from [From Autoencoder to Beta-VAE (lilianweng.github.io)](lilianweng.github.io).]

Denoising Autoencoders and BART

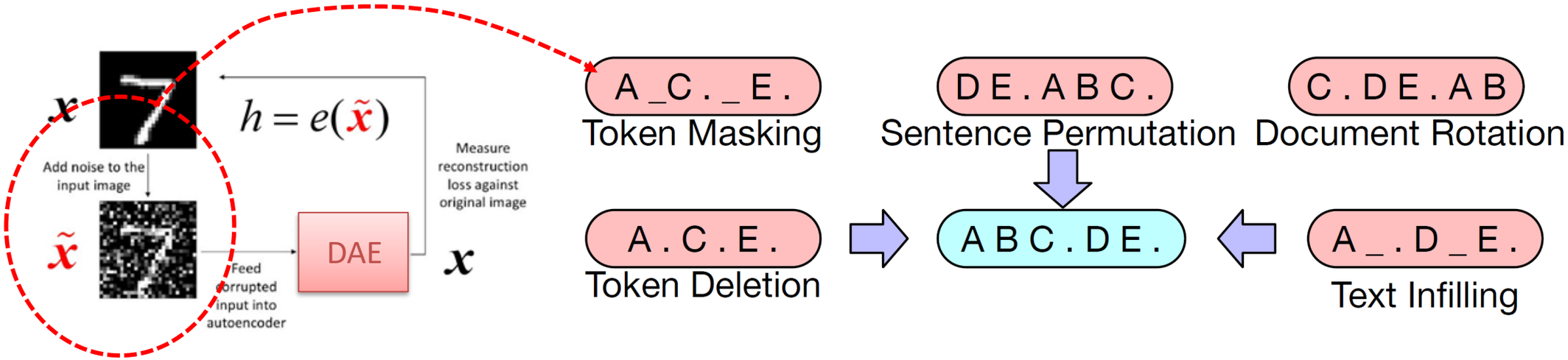
$$L\left(\boldsymbol{x}, d\left(e(\tilde{\boldsymbol{x}})\right)\right), \quad h = e(\tilde{\boldsymbol{x}})$$

Corrupted copy of $\boldsymbol{x}$

Autoencoders have to undo this corruption beyond simply coping the input.

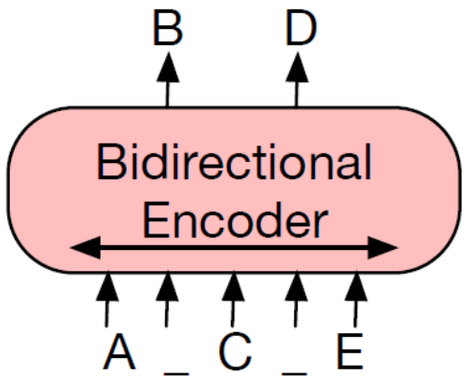
1. A training sample is sampled from the training data.
2. A corrupted version of the sample $\boldsymbol{x}$ is drawn from some noise model $\mathcal{M}_d\left(\tilde{\boldsymbol{x}} \middle| \boldsymbol{x}\right)$.
3. Pairs are used as a training sample to estimate the reconstruction distribution:

$$p_{\text{reconstruction}}\left(\boldsymbol{x} \middle| \tilde{\boldsymbol{x}}\right) = p_{\text{decoder}}\left(\boldsymbol{x} \middle| \boldsymbol{h}\right) = p_{\text{decoder}}\left(\boldsymbol{x} \middle| \boldsymbol{e}(\tilde{\boldsymbol{x}})\right)$$

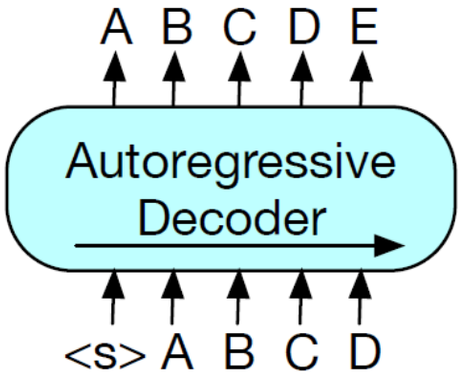


[Extracted from [1910.13461]; slightly modified for explanation purposes.]

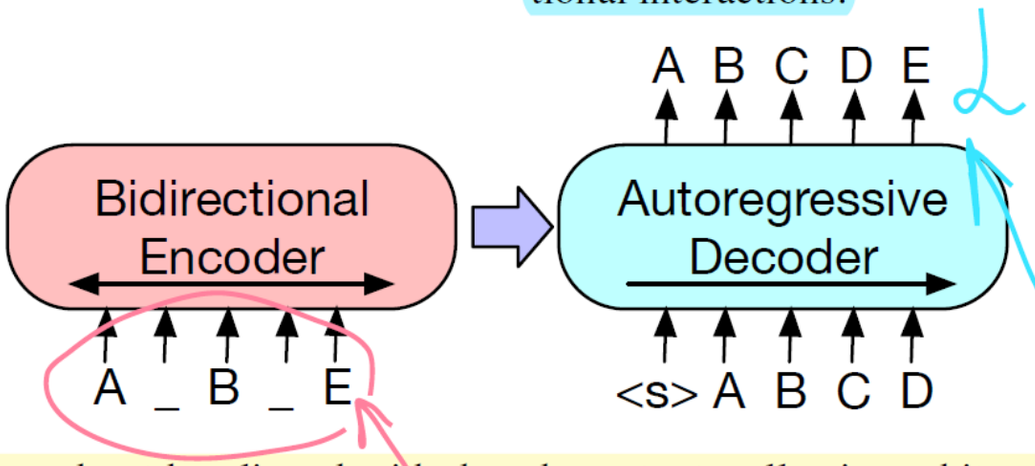
Denoising Autoencoders and BART



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

RAG-Sequence and RAG-Token models

- **RAG-Sequence:** this RAG variation uses the same document to predict each target token and thus generate the complete sequence in each output caption.
  - The retrieved document $\mathbf{d}(\boldsymbol{z})$ is treated as a single latent variable that is afterwards marginalized to get a seq2seq probability $\boldsymbol{p}(\boldsymbol{y}|\boldsymbol{x})$ via top *k* approximation (MIPS).
  - To reveal the whole output sequence $\boldsymbol{y}$ by a single latent factor; in practice we follow the pipeline below and then mathematically associate the components obtained:
    1. We use the retriever p_η to obtain the top *k* documents.
    2. We pass them $\left\{\mathbf{d}(\boldsymbol{z}_i)\right\}_{i=1}^{k}$ to the generator p_θ.
- We approximate: $p(\boldsymbol{y} \mid \boldsymbol{x}) \approx \sum_{\boldsymbol{z}} p_\eta(\boldsymbol{z} \mid \boldsymbol{x}) \prod_{i=1}^{N} p_\theta(\boldsymbol{y}_i \mid \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}_{1:i\text{-}1})$

RAG-Sequence and RAG-Token models

- RA ... ment to ... plete sec ...
  - ... ent ... 2seq
  - ... tent factor; in practice we follow the pipeline below and then mathematically associate the components obtained:
    1. We use the retriever p_η to obtain the top k documents.
    2. We pass them $\{\mathbf{d}(\mathbf{z}_i)\}_{i=1}^{k}$ to the generator p_θ.

Retrieval and generation probabilities

To compute $p(\boldsymbol{y} \mid \boldsymbol{x})$ we start by computing the retrieval probability and then the generation probability. The latter $p_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z})$ though by applying product rule can be written/interpreted as the probability of getting parts of the output sequence $\boldsymbol{y}$ given a latent document (an estimate) $\boldsymbol{z}$:

$$\prod_{i=1}^{N} p_\theta(\boldsymbol{y}_i \mid \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}_{1:i-1})$$

- We approximate: $p(\boldsymbol{y} \mid \boldsymbol{x}) \approx \sum_{\boldsymbol{z}} p_\eta(\boldsymbol{z} \mid \boldsymbol{x}) \prod_{i=1}^{N} p_\theta(\boldsymbol{y}_i \mid \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}_{1:i-1})$

RAG-Sequence and RAG-Token models

Retrieval and generation probabilities

To compute $p(\boldsymbol{y} \mid \boldsymbol{x})$ we start by computing the retrieval probability and then the generation probability. The latter $p_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z})$ though by applying product rule can be written/interpreted as the probability of getting parts of the output sequence $\boldsymbol{y}$ given a latent document (an estimate) $\boldsymbol{z}$:

$$\prod_{i=1}^{N} p_\theta(\boldsymbol{y}_i \mid \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}_{1:i\text{-}1})$$

Interpreting the generation probability

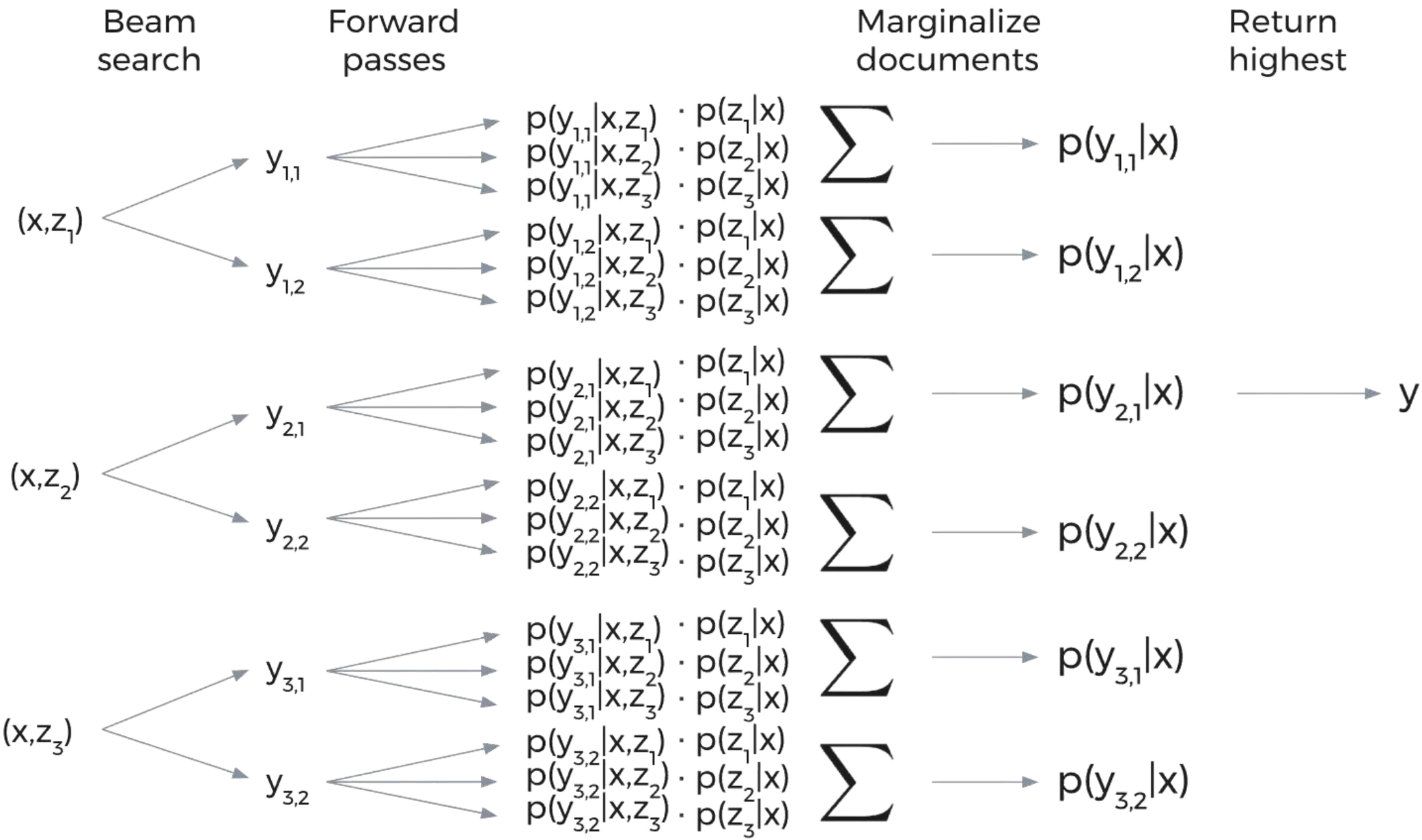
To express $p(\boldsymbol{y} \mid \boldsymbol{x})$ by also incorporating the interpretation of the generation probability given above we can further expand the steps of its approximation as:

$$p(\boldsymbol{y} \mid \boldsymbol{x}) \approx \sum_{\boldsymbol{z}} p_\eta(\boldsymbol{z} \mid \boldsymbol{x}) p_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}) = \sum_{\boldsymbol{z}} p_\eta(\boldsymbol{z} \mid \boldsymbol{x}) \prod_{i=1}^{N} p_\theta(\boldsymbol{y}_i \mid \boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}_{1:i\text{-}1})$$

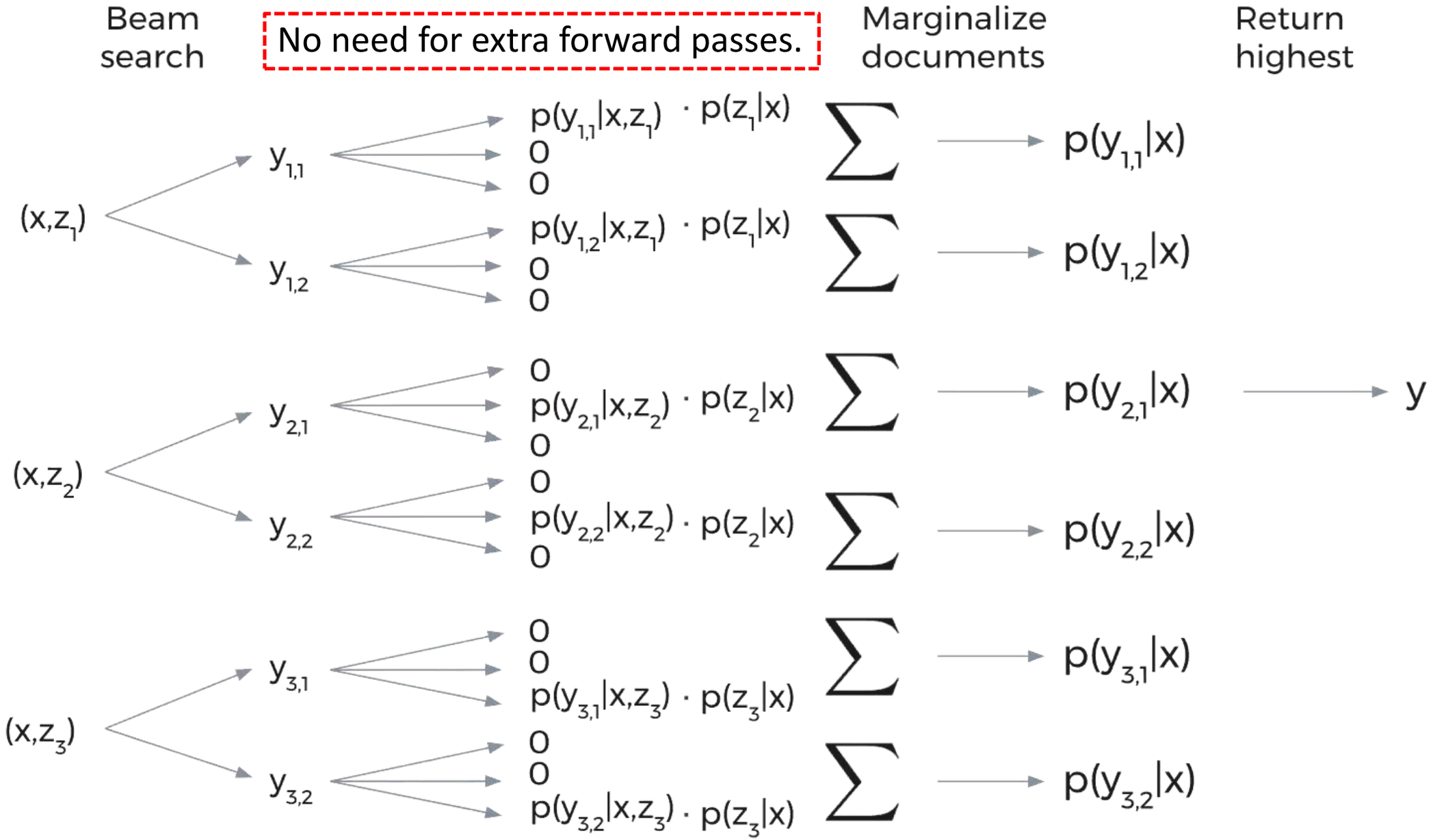
RAG-Sequence and RAG-Token models

- **RAG-Sequence:** this RAG variation uses the same document to predict each target token and thus generate the complete sequence in each output caption.
  - For decoding at test time, we run beam search for every document $\boldsymbol{z}$, scoring each hypothesis using the Language Model obtained by the generator p_θ, $p_\theta(y_i|x,z,y_{1:i-1})$.
  - This yields a set of hypotheses Y maybe occurring only in particular documents' beams. To estimate the probability of a hypothesis we may follow two decoding procedures:
    - *Thorough Decoding:* run an extra forward pass for all $\boldsymbol{z}$ not appeared in the beam, with $p_\theta(y_i|x,z,y_{1:i-1})$ generate $p_\eta(z|x)$ and sum across beams for the marginals.
    - *Fast Decoding:* $p_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z}_i) \approx 0$ ($\boldsymbol{y}$ not generated from $\boldsymbol{x}$, $\boldsymbol{z}_i$)

RAG-Sequence and RAG-Token models



RAG-Sequence and RAG-Token models



$p_\theta(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{z_i}) \approx 0$ ($\boldsymbol{y}$ not generated from $\boldsymbol{x}$, $\boldsymbol{z_i}$)

RAG-Sequence and RAG-Token models

- **RAG-Token:** this RAG variation draws a different document $\boldsymbol{z}$ to predict each target token and marginalize accordingly. In this context the generator can combine content from several documents to produce the output caption $\boldsymbol{y}$.
  - The top *k* retrieved documents $\{\mathbf{d}(\boldsymbol{z}_i)\}_{i=1}^{k}$ are obtained.
  - To reveal the whole output sequence $\boldsymbol{y}$; we can follow the similar pipeline below and then mathematically associate the components obtained:
    1. We use the retriever p_η to obtain the top *k* documents.
    2. We pass them $\{\mathbf{d}(\boldsymbol{z}_i)\}_{i=1}^{k}$ to the generator p_θ.
    3. We marginalize <u>per token we generate</u>.
- We approximate: $p(\boldsymbol{y} \mid \boldsymbol{x}) \approx \prod_{i=1}^{N} \sum_{\boldsymbol{z}_i} p_\eta(\boldsymbol{z}_i \mid \boldsymbol{x}) p_\theta(\boldsymbol{y}_i \mid \boldsymbol{x}, \boldsymbol{z}_i, \boldsymbol{y}_{1:i\text{-}1})$

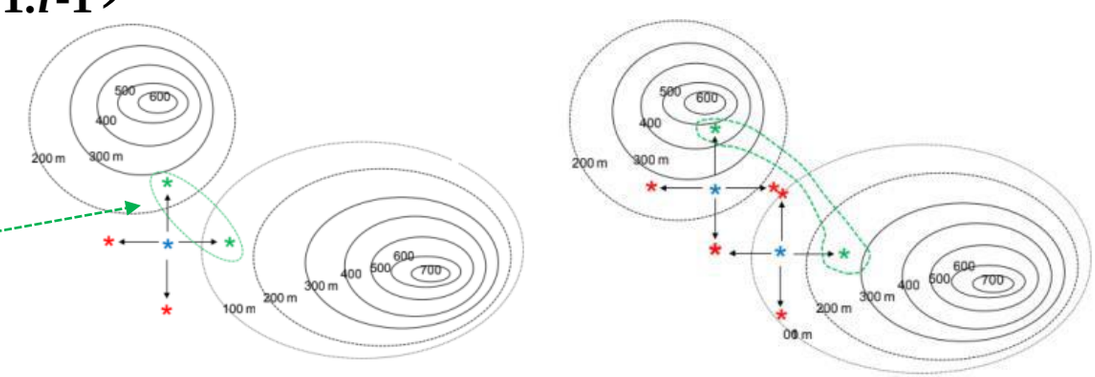
RAG-Sequence and RAG-Token models

- **RAG-Token:** this RAG variation draws a different document $\boldsymbol{z}$ to predict each target token and marginalize accordingly. In this context the generator can combine content from several documents to produce the output caption $\boldsymbol{y}$.
  - The decoding follows the typical, autoregressive seq2seq generation pipeline, with transition prob. $p'_\theta(\boldsymbol{y_i} \mid \boldsymbol{x}, \boldsymbol{y_{1:i\text{-}1}})$ given by summing over the top *k* representations:

$$p'_\theta(\boldsymbol{y_i} \mid \boldsymbol{x}, \boldsymbol{y_{1:i\text{-}1}}) = \sum_{\boldsymbol{z_i}} p_\eta(\boldsymbol{z_i} \mid \boldsymbol{x}) p_\theta(\boldsymbol{y_i} \mid \boldsymbol{x}, \boldsymbol{z_i}, \boldsymbol{y_{1:i\text{-}1}})$$

  - We plug $p'_\theta(\boldsymbol{y_i} \mid \boldsymbol{x}, \boldsymbol{y_{1:i\text{-}1}})$ into a standard beam decoder.

In local beam search we will move towards states $\{\boldsymbol{x_i}\}_i$ with the highest values (in ascent) of a metric V_i.



Training Practicalities and formulas

- The document encoder $\mathbf{d}$ is trained during pre-process phase <u>once</u> and using a large document collection, like *Wikipedia*, if possible domain-related.
- Within the retriever p_η (non-parametric memory):
  - We encode each document $\boldsymbol{x}$ into a dense representation.
  - We apply Maximum Inner Product Search sim. algorithm.
  - We pass the documents with the maximum dot product to the seq2seq generator.
- Within the generator p_θ (parametric memory):
  - Per document $\mathbf{d}(\boldsymbol{x})$ we get a prediction of the generation.
  - We marginalize *accordingly*, to get an output sequence $\boldsymbol{y}$, w.r.t. to what type of RAG model we are using. Decoding process is also adjusted.

Training Practicalities and formulas

- We treat questions-answers as input-output text pairs $(\boldsymbol{x}, \boldsymbol{t})$ and then train either RAG-Sequence or RAG-token by directly minimizing the negative marginal log-likelihood of generating output sequences $\boldsymbol{y}$ on input sequences $\boldsymbol{x}$. If $\mathcal{D} = \{\boldsymbol{x}_j, \boldsymbol{t}_j\}_j$ is the complete dataset; we aim to minimize:

$$l_{\text{cross}}(\boldsymbol{x}, \boldsymbol{t}; \mathbf{w}) = -\log p(\boldsymbol{y} \mid \boldsymbol{x}; \mathbf{w}) \ \forall \boldsymbol{x}, \boldsymbol{t}$$

$$E(\mathbf{w}) = \sum_j l_{\text{cross}}(\boldsymbol{x}_j, \boldsymbol{t}_j; \mathbf{w}) = \sum_j -\log p(\boldsymbol{y}_j \mid \boldsymbol{x}_j; \mathbf{w})$$

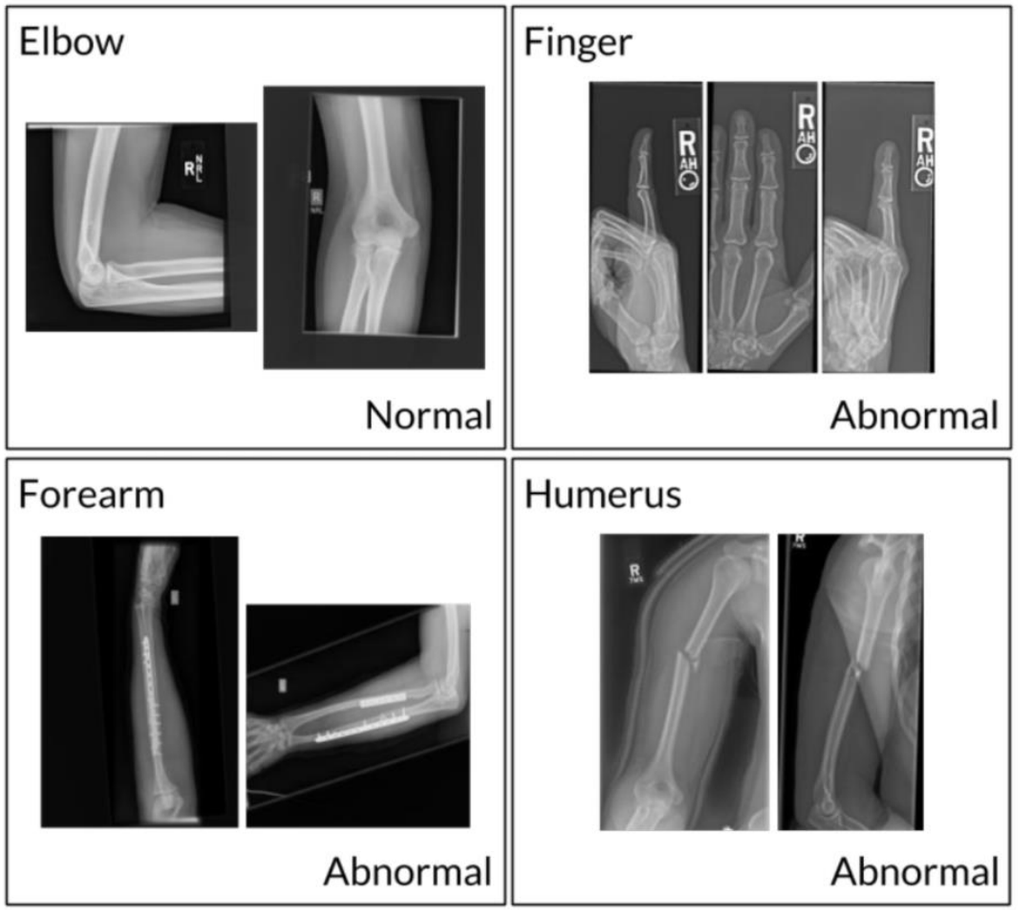
- Adaptive Moments (Adam) optimizer:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \varepsilon \frac{\mathbf{v}^{(t)}}{\delta + \sqrt{\mathbf{r}^{(t)}}}, \ \delta, \ \varepsilon \in \mathbb{R}^+$$

$$\mathbf{v}^{(t+1)} = \rho_1 \mathbf{v}^{(t)} + (1-\rho_1)\mathbf{g}^{(t)}, \ \mathbf{r}^{(t+1)} = \rho_2 \mathbf{r}^{(t)} + (1-\rho_2)\left(\mathbf{g}^{(t)}\right)^2, \ \rho_1, \ \rho_2 \in \mathbb{R}^+$$

(Combines advantages of RMSProp and momentum velocity.)

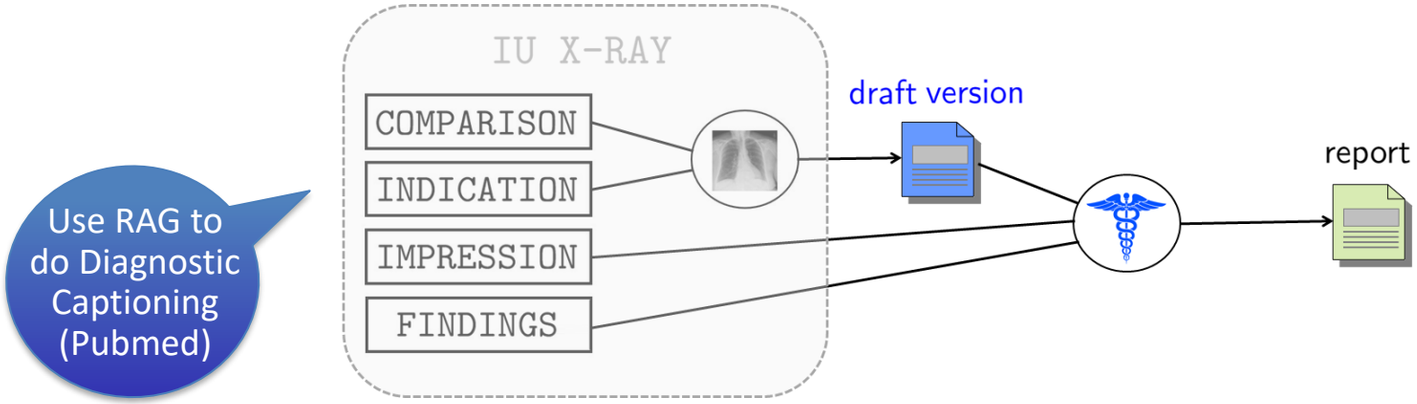
Diagnostic captioning



FINDINGS: No change. No visible active cardiopulmonary disease. Both lungs remain clear and expanded. Heart and pulmonary XXXX are normal. No change in the large hiatus hernia.

SYSTEM TAGS: hiatus hernia / Hernia, Hiatus

HUMAN TAGS: Hernia, Hiatal/ large



Procedure P_1: Preprocessing and caption generation

Input: an image $\vec{x} \in \mathcal{D}$

Output: a pair $\vec{x}, \vec{t}$ of the image $\vec{x} \in \mathcal{D}$ and its associated target caption $\vec{t} \in \mathcal{C}$

GOLD , CAPTION

if $class(\vec{x}) = c_3$ set the caption $\vec{t} = fields.FINDINGS(\vec{x})$

if $class(\vec{x}) = c_4$ set the caption $\vec{t} = fields.IMPRESSION(\vec{x})$

if $class(\vec{x}) = c_5$ set the caption $\vec{t} = fields.IMPRESSION(\vec{x})$ + " " + $fields.FINDINGS(\vec{x})$

return $\vec{x}, \vec{t}$

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \varepsilon \frac{\mathbf{v}^{(t)}}{\delta + \sqrt{\mathbf{r}^{(t)}}},\ \delta,\ \varepsilon \in \mathbb{R}^+ \text{ w.r.t. } \mathbf{v}^{(t+1)} = \rho_1 \mathbf{v}^{(t)} + (1-\rho_1)\mathbf{g}^{(t)},\ \mathbf{r}^{(t+1)} = \rho_2 \mathbf{r}^{(t)} + (1-\rho_2)\left(\mathbf{g}^{(t)}\right)^2,\ \rho_1, \rho_2 \in \mathbb{R}^+$$

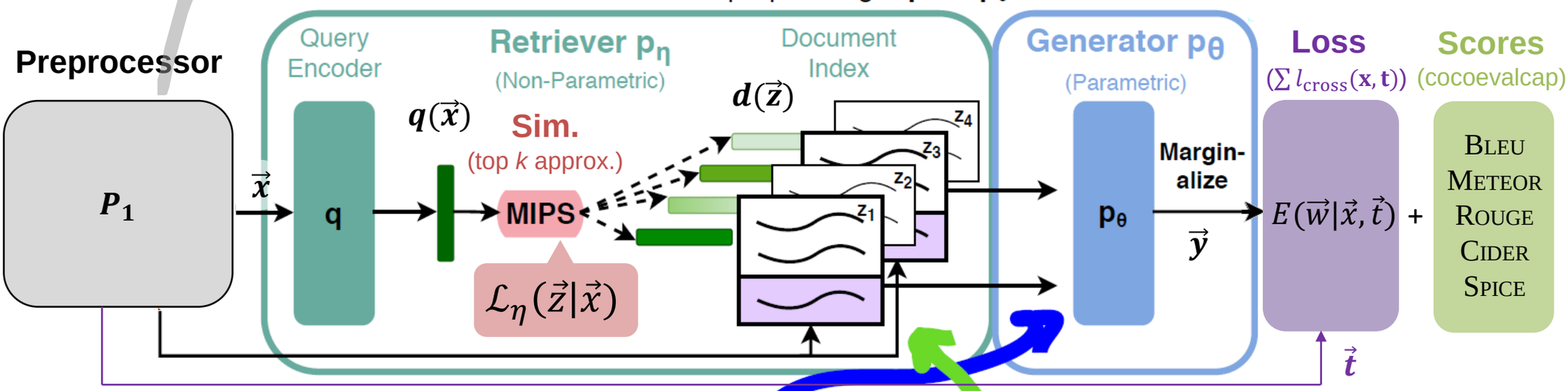


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

[Extracted from 2005.11401; appropriately modified for DC purposes.]

Happy to take questions!

(geomos@kth.se)

Image source: Getty Images/iStockphoto © Frender