UMass · CS685 | Advanced Natural Language Processing (2020)

CS685 (2020)· 课程资料包 @ShowMeAI

视频
中英双语字幕

课件
一键打包下载

笔记
官方笔记翻译

代码
作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1BL411t7RV

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/umass-cs685

NLP 语言模型 问答系统 文本生成 BERT 语义解析 GPT-3 知识推理 模型蒸馏 迁移学习 transformer 注意力机制

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets· 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者?** 回复 [添砖加瓦]

Retrieval-augmented language models

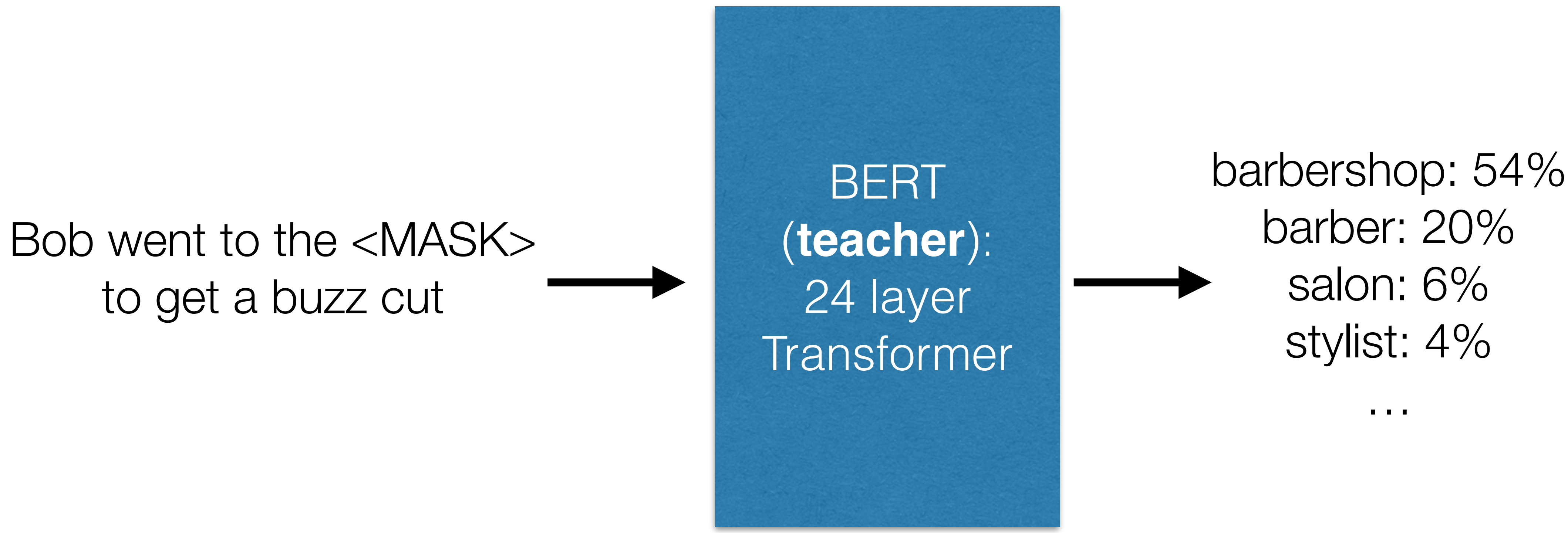
CS 685, Fall 2020

Advanced Natural Language Processing

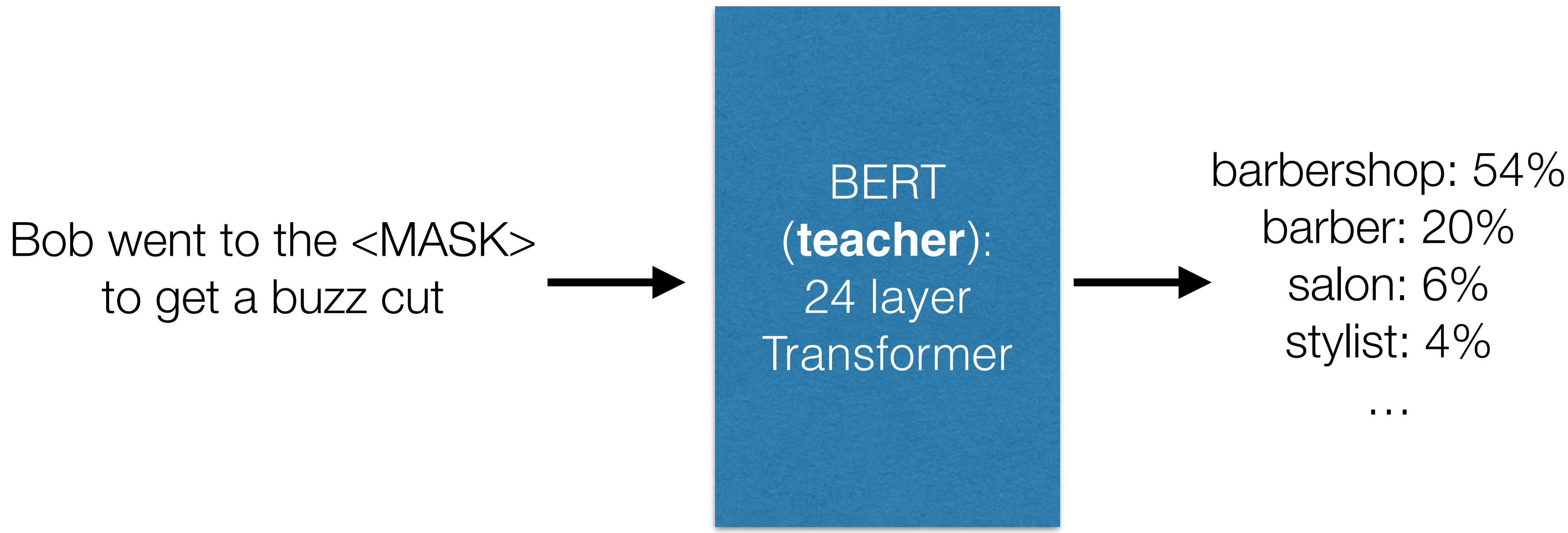
Mohit Iyyer

College of Information and Computer Sciences

University of Massachusetts Amherst


Bob went to the <MASK>
to get a buzz cut
BERT
(**teacher**):
24 layer
Transformer
barbershop: 54%
barber: 20%
salon: 6%
stylist: 4%
...

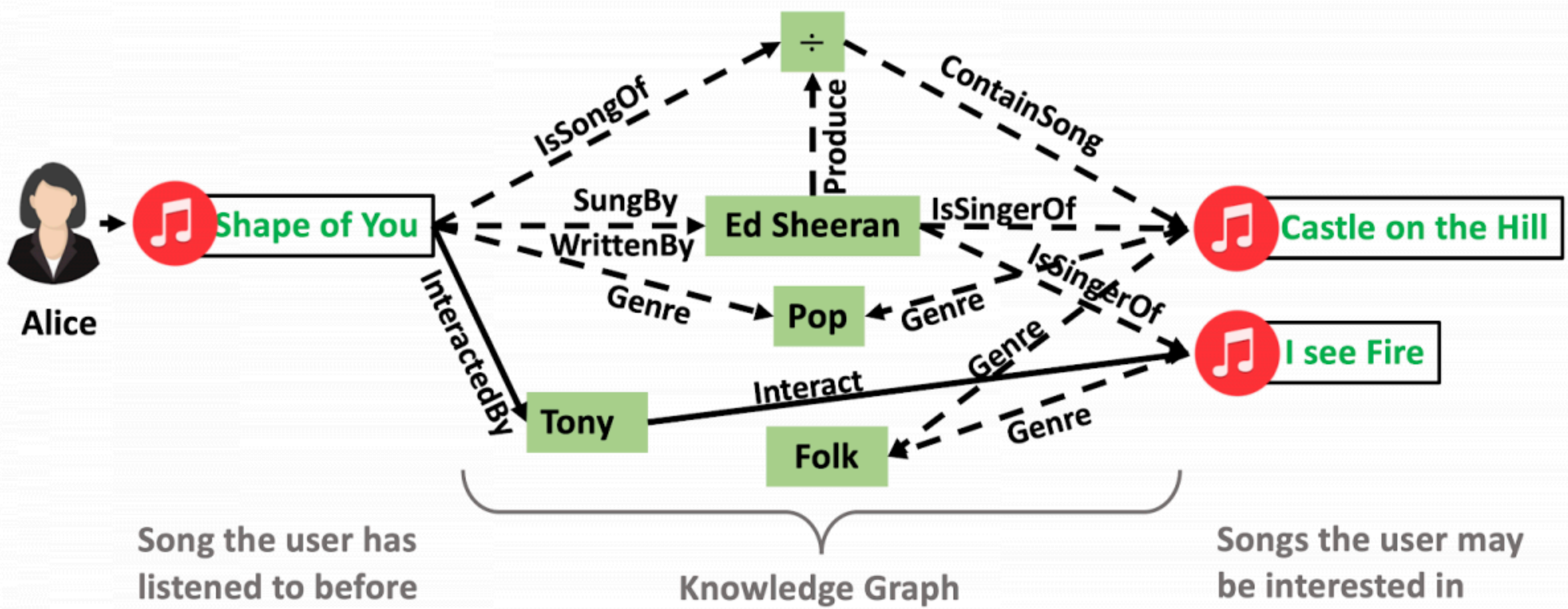
World knowledge is *implicitly* encoded in BERT's parameters! (e.g., that barbershops are places to get buzz cuts)



In these language models, the learned world knowledge is stored *implicitly* in the parameters of the underlying neural network. This makes it difficult to determine what knowledge is stored in the network and where. Furthermore, storage space is limited by the size of the network—to capture more world knowledge, one must train ever-larger networks, which can be prohibitively slow or expensive.

Guu et al., 2020 (“REALM”)

One option: condition predictions on explicit *knowledge graphs*



Wang et al., 2019

Pros / cons

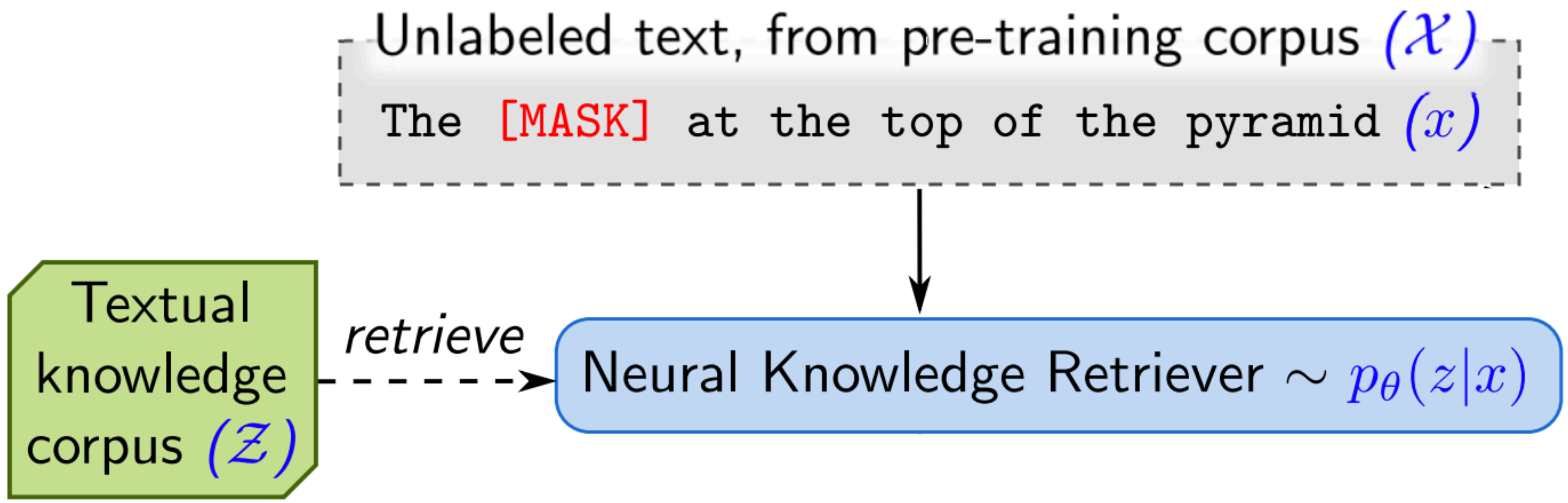
- Explicit graph structure makes KGs easy to navigate
- Knowledge graphs are expensive to produce at scale
- Automatic knowledge graph induction is an open research problem
- Knowledge graphs struggle to encode complex relations between entities

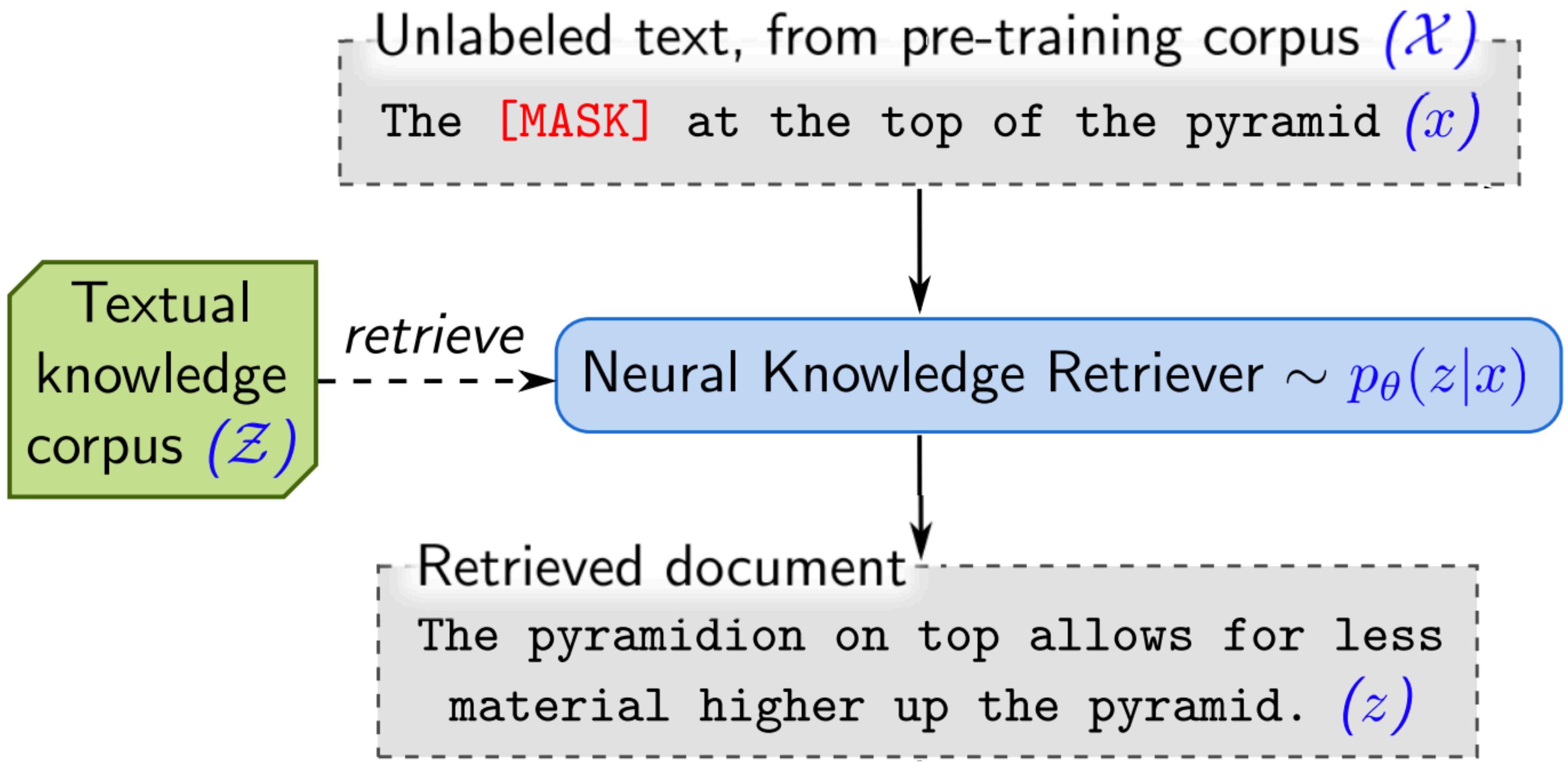
Another source of knowledge: unstructured text!

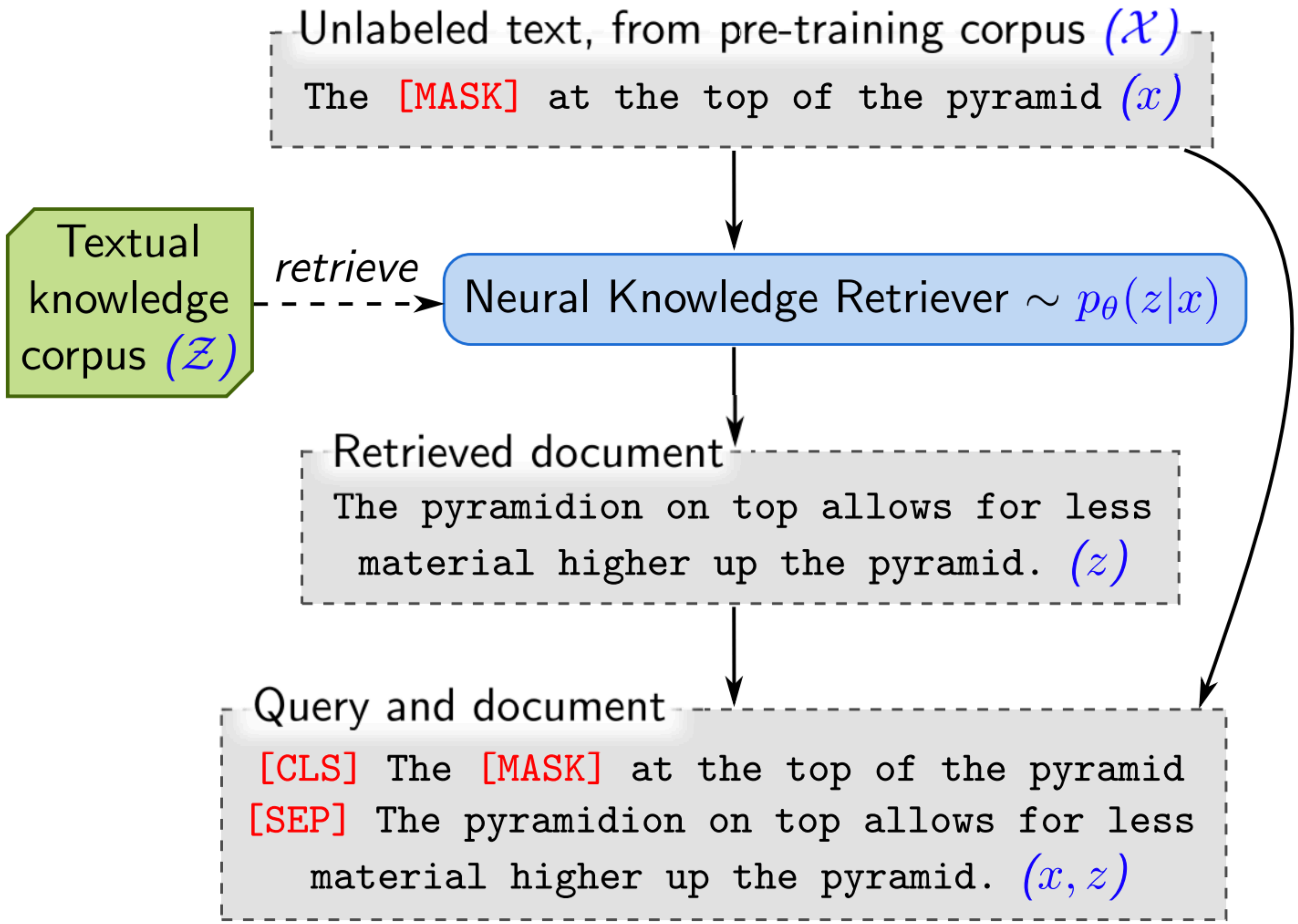
- Readily available at scale, requires no processing
- We have powerful methods of encoding semantics (e.g., BERT)
- However, these methods don’t really work with larger units of text (e.g., books)
- Extracting relevant information from unstructured text is more difficult than it is with KGs

Unlabeled text, from pre-training corpus $(\mathcal{X})$

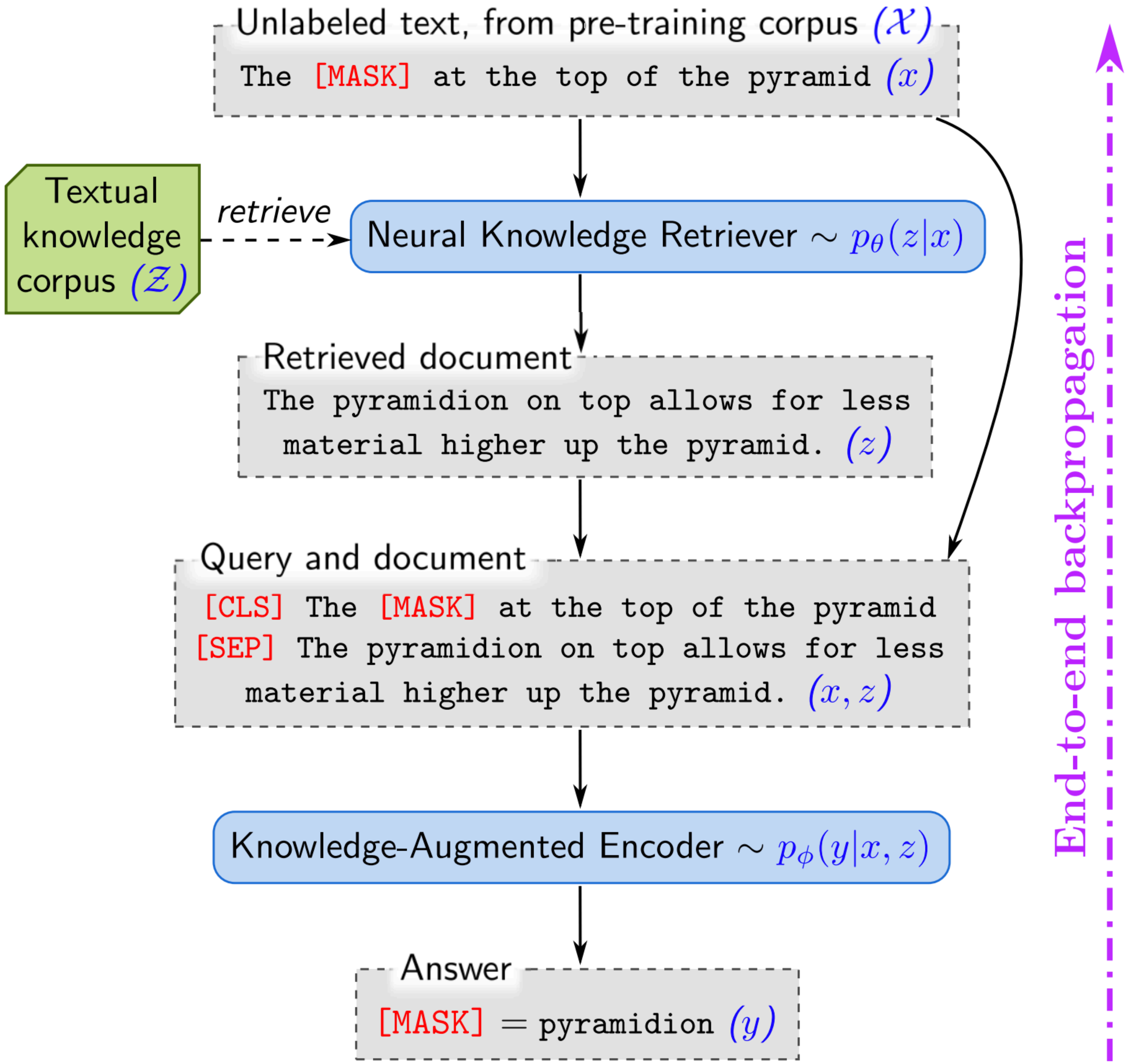
`The [MASK] at the top of the pyramid` (x)


Unlabeled text, from pre-training corpus ($\mathcal{X}$)
The [MASK] at the top of the pyramid (x)
Textual knowledge corpus ($\mathcal{Z}$)
retrieve
Neural Knowledge Retriever $\sim p_\theta(z|x)$

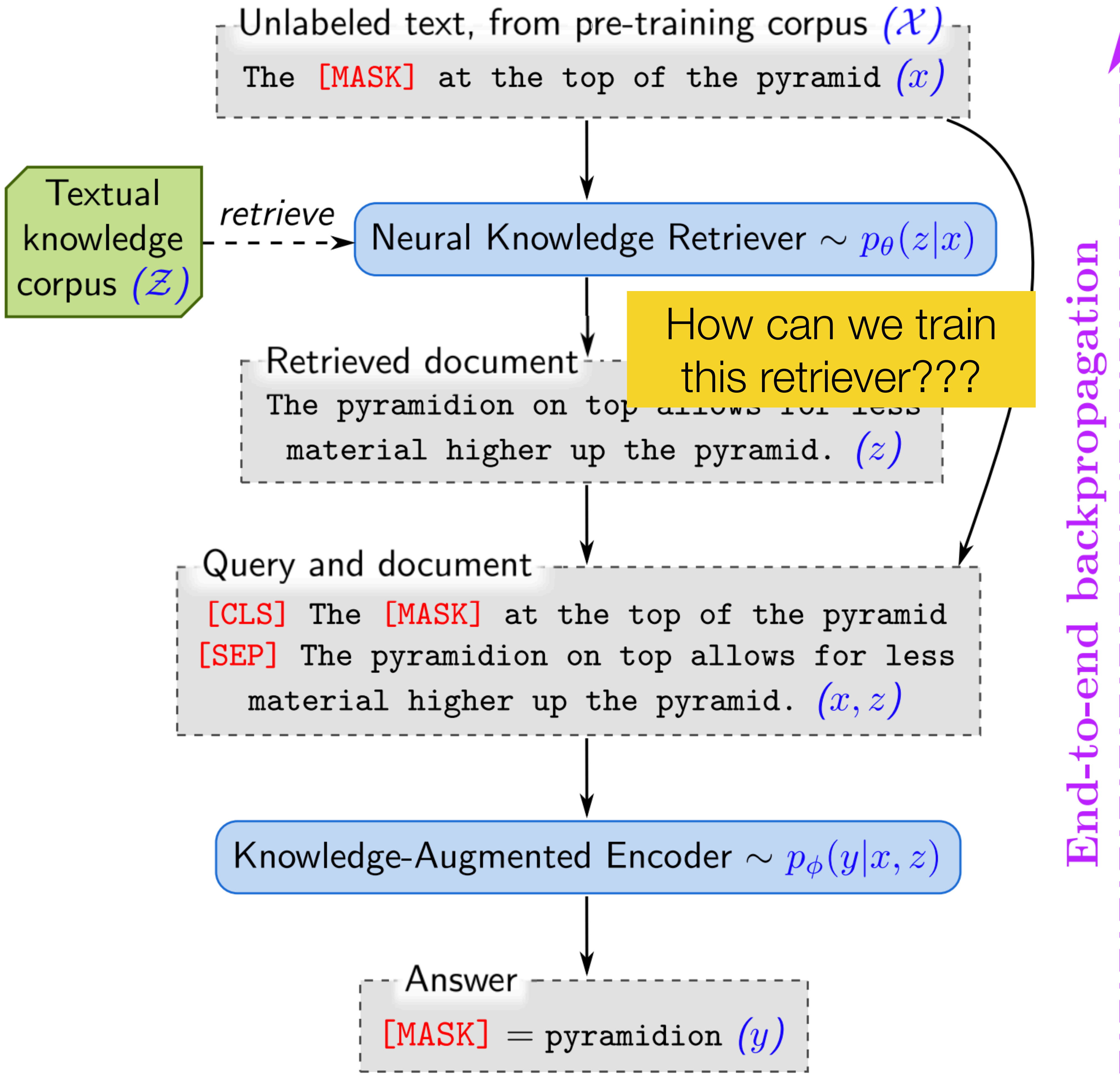

Unlabeled text, from pre-training corpus ($\mathcal{X}$)
The [MASK] at the top of the pyramid (x)
Textual knowledge corpus ($\mathcal{Z}$)
retrieve
Neural Knowledge Retriever $\sim p_\theta(z|x)$
Retrieved document
The pyramidion on top allows for less material higher up the pyramid. (z)



Unlabeled text, from pre-training corpus $(\mathcal{X})$
The [MASK] at the top of the pyramid (x)
Textual knowledge corpus $(\mathcal{Z})$
retrieve
Neural Knowledge Retriever $\sim p_\theta(z|x)$
Retrieved document
The pyramidion on top allows for less material higher up the pyramid. (z)
Query and document
[CLS] The [MASK] at the top of the pyramid [SEP] The pyramidion on top allows for less material higher up the pyramid. (x, z)



Unlabeled text, from pre-training corpus $(\mathcal{X})$
The [MASK] at the top of the pyramid (x)
Textual knowledge corpus $(\mathcal{Z})$
retrieve
Neural Knowledge Retriever $\sim p_\theta(z|x)$
Retrieved document
The pyramidion on top allows for less material higher up the pyramid. (z)
Query and document
[CLS] The [MASK] at the top of the pyramid [SEP] The pyramidion on top allows for less material higher up the pyramid. (x, z)
Knowledge-Augmented Encoder $\sim p_\phi(y|x, z)$
Answer
[MASK] = pyramidion (y)
End-to-end backpropagation


Unlabeled text, from pre-training corpus $(\mathcal{X})$
The [MASK] at the top of the pyramid (x)
Textual knowledge corpus $(\mathcal{Z})$
retrieve
Neural Knowledge Retriever $\sim p_\theta(z|x)$
How can we train this retriever???
Retrieved document
The pyramidion on top
material higher up the pyramid. (z)
Query and document
[CLS] The [MASK] at the top of the pyramid
[SEP] The pyramidion on top allows for less
material higher up the pyramid. (x, z)
Knowledge-Augmented Encoder $\sim p_\phi(y|x, z)$
Answer
[MASK] = pyramidion (y)
End-to-end backpropagation

REALM decomposes $p(y \mid x)$ into two steps: *retrieve*, then *predict*. Given an input x, we first retrieve possibly helpful documents z from a knowledge corpus $\mathcal{Z}$. We model this as a sample from the distribution $p(z \mid x)$. Then, we condition on both the retrieved z and the original input x to generate the output y—modeled as $p(y \mid z, x)$. To obtain the overall likelihood of generating y, we treat z as a latent variable and marginalize over all possible documents z, yielding

$$p(y \mid x) = \sum_{z \in \mathcal{Z}} p(y \mid z, x)\, p(z \mid x)$$

REALM decomposes $p(y \mid x)$ into two steps: *retrieve*, then *predict*. Given an input x, we first retrieve possibly helpful documents z from a knowledge corpus $\mathcal{Z}$. We model this as a sample from the distribution $p(z \mid x)$. Then, we condition on both the retrieved z and the original input x to generate the output y—modeled as $p(y \mid z, x)$. To obtain the overall likelihood of generating y, we treat z as a latent variable and marginalize over all possible documents z, yielding

Knowledge-augmented encoder

$$p(y \mid x) = \sum_{z \in \mathcal{Z}} p(y \mid z, x)\, p(z \mid x)$$

Neural knowledge retriever

Knowledge Retriever The retriever is defined using a dense inner product model:

$$p(z \mid x) = \frac{\exp f(x, z)}{\sum_{z'} \exp f(x, z')},$$
$$f(x, z) = \texttt{Embed}_{\texttt{input}}(x)^\top \texttt{Embed}_{\texttt{doc}}(z),$$

where $\texttt{Embed}_{\texttt{input}}$ and $\texttt{Embed}_{\texttt{doc}}$ are embedding functions that map x and z respectively to d-dimensional vectors. The *relevance score* $f(x, z)$ between x and z is defined as the inner product of the vector embeddings. The retrieval distribution is the softmax over all relevance scores.

Embed function is just BERT!

$$\texttt{join}_{\texttt{BERT}}(x) = \texttt{[CLS]}\,x\,\texttt{[SEP]}$$

$$\texttt{join}_{\texttt{BERT}}(x_1, x_2) = \texttt{[CLS]}\,x_1\,\texttt{[SEP]}\,x_2\,\texttt{[SEP]}$$

$$\texttt{Embed}_{\texttt{input}}(x) = \mathbf{W}_{\texttt{input}}\mathrm{BERT}_{\mathrm{CLS}}(\texttt{join}_{\texttt{BERT}}(x))$$

$$\texttt{Embed}_{\texttt{doc}}(z) = \mathbf{W}_{\texttt{doc}}\mathrm{BERT}_{\mathrm{CLS}}(\texttt{join}_{\texttt{BERT}}(z_{\mathrm{title}}, z_{\mathrm{body}}))$$

Knowledge-Augmented Encoder Given an input x and a retrieved document z, the knowledge-augmented encoder defines $p(y \mid z, x)$. We join x and z into a single sequence that we feed into a Transformer (distinct from the one used in the retriever).

$$p(y \mid z, x) = \prod_{j=1}^{J_x} p(y_j \mid z, x)$$

$$p(y_j \mid z, x) \propto \exp\left(w_j^\top \mathtt{BERT}_{\mathtt{MASK}(j)}(\mathtt{join}_{\mathtt{BERT}}(x, z_{\text{body}}))\right)$$

where $\mathtt{BERT}_{\mathtt{MASK}(j)}$ denotes the Transformer output vector corresponding to the j^{th} masked token, J_x is the total number of [MASK] tokens in x, and w_j is a learned word embedding for token y_j.

Isn't training the retriever extremely expensive?

The key computational challenge is that the marginal probability $p(y \mid x) = \sum_{z \in \mathcal{Z}} p(y \mid x, z)\, p(z \mid x)$ involves a summation over all documents z in the knowledge corpus $\mathcal{Z}$. We approximate this by instead summing over the top k documents with highest probability under $p(z \mid x)$—this is reasonable if most documents have near zero probability.

Imagine if your knowledge corpus was every article in Wikipedia… this would be super expensive without the approximation

Maximum inner product search (MIPS)

- Algorithms that *approximately* find the top-*k* documents
- Scales *sub-linearly* with the number of documents (both time and storage)
    - Shrivastava and Li, 2014 (“Asymmetric LSH…”)
- Requires precomputing the BERT embedding of every document in the knowledge corpus and then building an index over the embeddings

Need to refresh the index!

- We are training the parameters of the retriever, i.e., the BERT architecture that produces $\mathbf{Embed_{doc}(z)}$

- If we precompute all of the embeddings, the search index becomes stale when we update the parameters of the retriever

- REALM solution: asynchronously refresh the index by re-embedding all docs after a few hundred training iterations

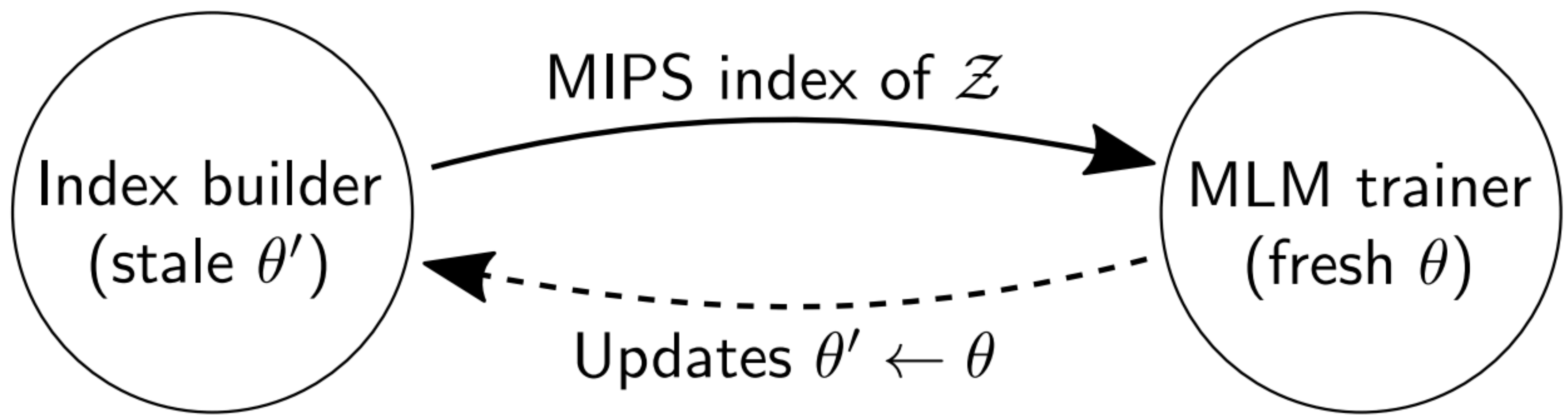


Figure 3. REALM pre-training with asynchronous MIPS refreshes.

Other tricks in REALM

- *Salient span masking*: mask out spans of text corresponding to named entities and dates

- *Null document*: always include an empty document in the top-*k* retrieved docs, allowing the model to rely on its implicit knowledge as well

Evaluation on *open-domain QA*

- Unlike SQuAD-style QA, in open-domain QA we are only given a question, not a supporting document that is guaranteed to contain the answer

- Open-domain QA generally has a large *retrieval* component, since the answer to any given question could occur anywhere in a large collection of documents

Name	Architectures	Pre-training	**NQ** (79k/4k)	**WQ** (3k/2k)	**CT** (1k /1k)	# params
BERT-Baseline (Lee et al., 2019)	Sparse Retr.+Transformer	BERT	26.5	17.7	21.3	110m
T5 (base) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	27.0	29.1	-	223m
T5 (large) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	29.8	32.2	-	738m
T5 (11b) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	34.5	37.4	-	11318m
DrQA (Chen et al., 2017)	Sparse Retr.+DocReader	N/A	-	20.7	25.7	34m
HardEM (Min et al., 2019a)	Sparse Retr.+Transformer	BERT	28.1	-	-	110m
GraphRetriever (Min et al., 2019b)	GraphRetriever+Transformer	BERT	31.8	31.6	-	110m
PathRetriever (Asai et al., 2019)	PathRetriever+Transformer	MLM	32.6	-	-	110m
ORQA (Lee et al., 2019)	Dense Retr.+Transformer	ICT+BERT	33.3	36.4	30.1	330m
Ours ($\mathcal{X}$ = Wikipedia, $\mathcal{Z}$ = Wikipedia)	Dense Retr.+Transformer	REALM	39.2	40.2	**46.8**	330m
Ours ($\mathcal{X}$ = CC-News, $\mathcal{Z}$ = Wikipedia)	Dense Retr.+Transformer	REALM	**40.4**	**40.7**	42.9	330m

Table 3. An example where REALM utilizes retrieved documents to better predict masked tokens. It assigns much higher probability (0.129) to the correct term, "`Fermat`", compared to BERT. (Note that the blank corresponds to 3 BERT wordpieces.)

	x:	An equilateral triangle is easily constructed using a straightedge and compass, because 3 is a ___ prime.		
(a)	BERT	$p(y = \text{“}\texttt{Fermat}\text{”} \mid x)$	$= 1.1 \times 10^{-14}$	(No retrieval.)
(b)	REALM	$p(y = \text{“}\texttt{Fermat}\text{”} \mid x, z)$	$= 1.0$	(Conditional probability with document $z =$“257 is ... a Fermat prime. Thus a regular polygon with 257 sides is constructible with compass ...”)
(c)	REALM	$p(y = \text{“}\texttt{Fermat}\text{”} \mid x)$	$= 0.129$	(Marginal probability, marginalizing over top 8 retrieved documents.)

Can retrieval-augmented LMs improve other tasks?

Nearest-neighbor machine translation

Test Input x	**Generated tokens** $\hat{y}_{1:i-1}$	**Representation** $q = f(x, \hat{y}_{1:i-1})$	**Target** y_i
J'ai été dans ma propre chambre.	*I have*		*?*

Khandelwal et al., 2020

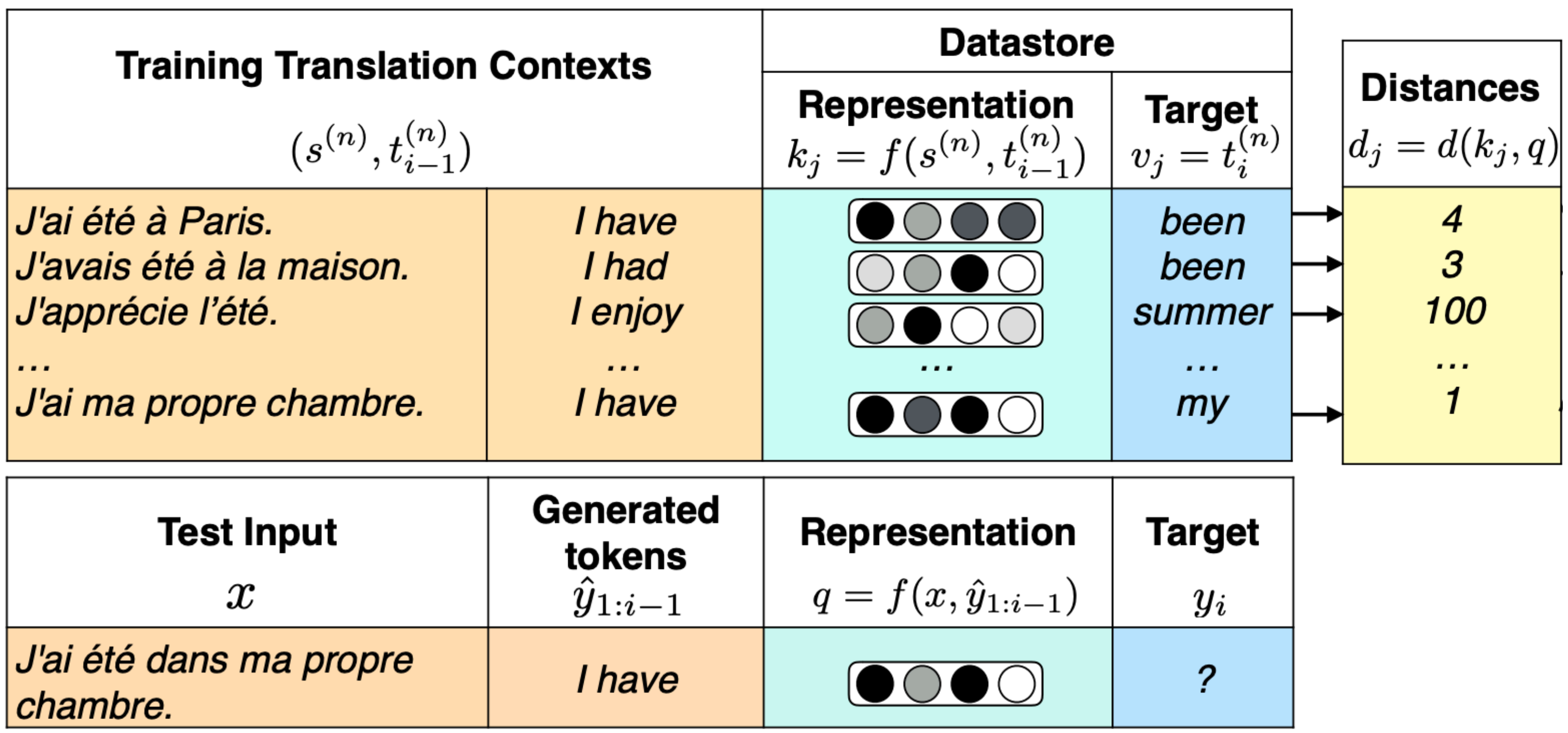
Nearest-neighbor machine translation

Training Translation Contexts $(s^{(n)}, t_{i-1}^{(n)})$		**Datastore** **Representation** $k_j = f(s^{(n)}, t_{i-1}^{(n)})$	**Target** $v_j = t_i^{(n)}$
J'ai été à Paris.	*I have*		*been*
J'avais été à la maison.	*I had*		*been*
J'apprécie l'été.	*I enjoy*		*summer*
…	…	…	…
J'ai ma propre chambre.	*I have*		*my*

Test Input x	**Generated tokens** $\hat{y}_{1:i-1}$	**Representation** $q = f(x, \hat{y}_{1:i-1})$	**Target** y_i
J'ai été dans ma propre chambre.	*I have*		*?*

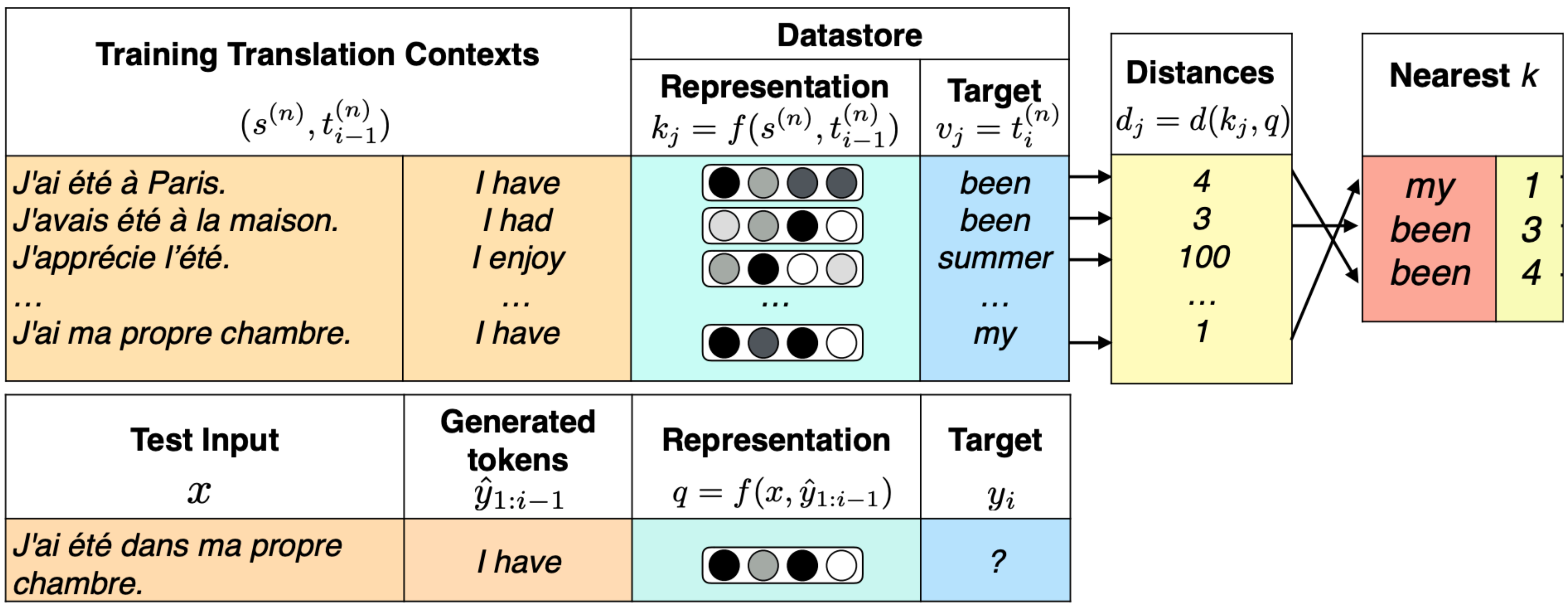
Khandelwal et al., 2020

Nearest-neighbor machine translation



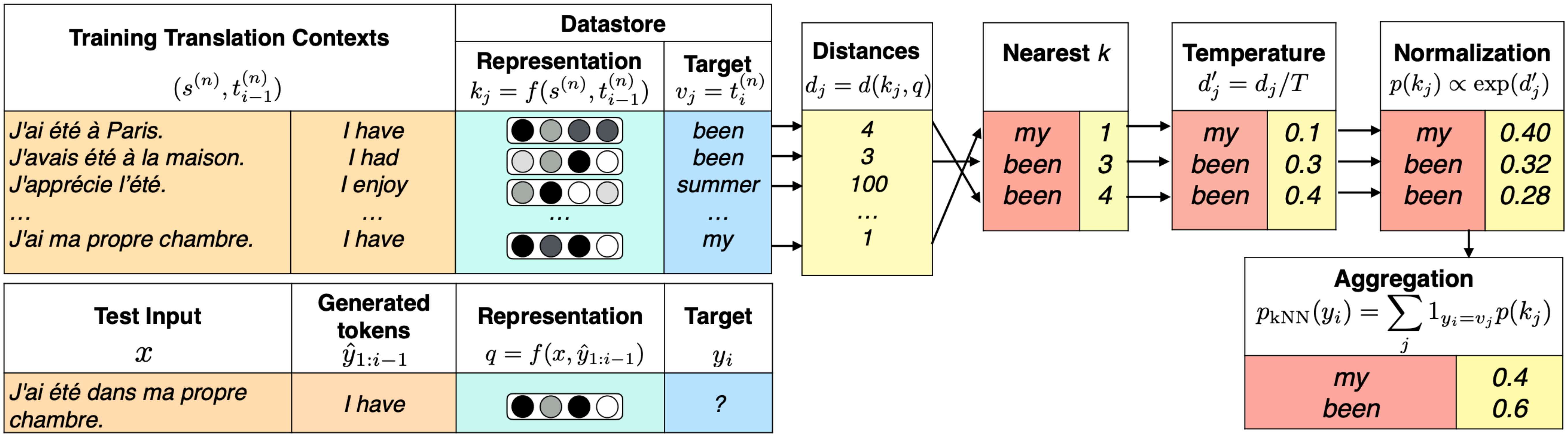
Khandelwal et al., 2020

Nearest-neighbor machine translation



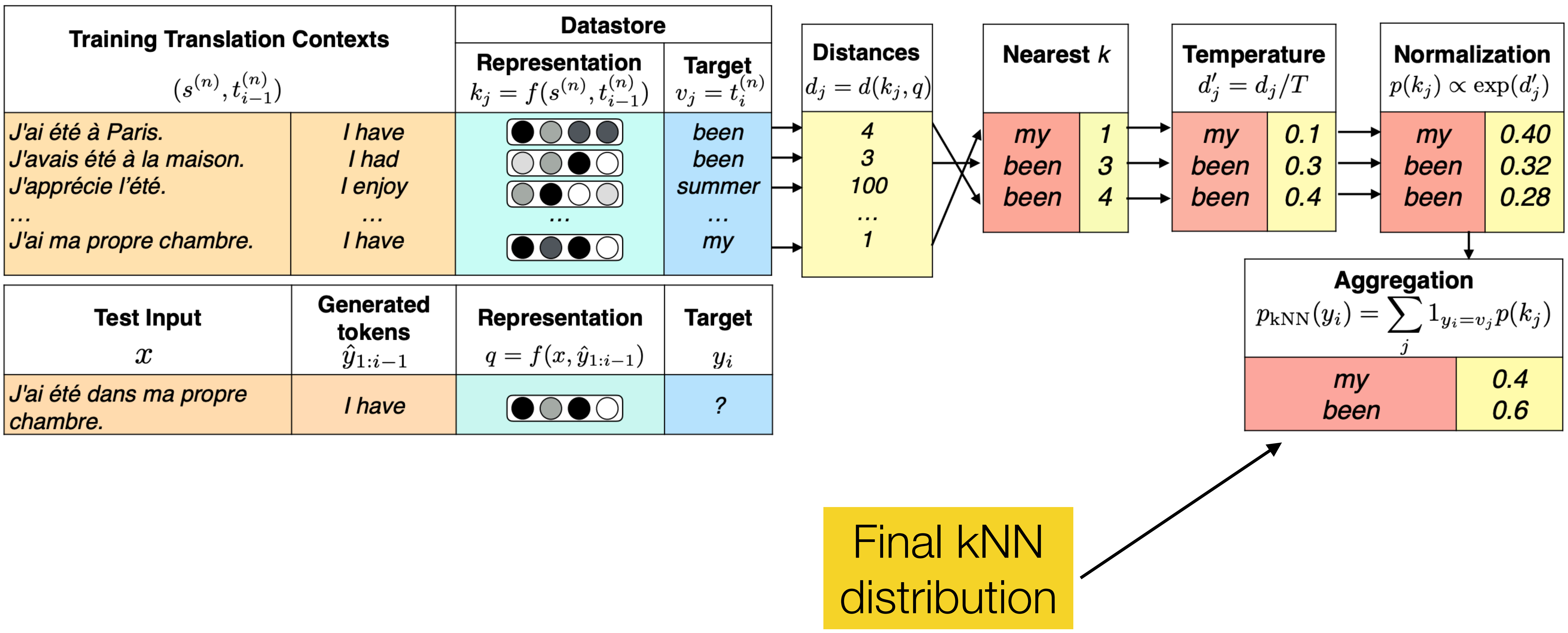
Khandelwal et al., 2020

Nearest-neighbor machine translation



Khandelwal et al., 2020

Nearest-neighbor machine translation



Khandelwal et al., 2020

Interpolate between kNN prediction and decoder's actual prediction

$$p(y_i|x, \hat{y}_{1:i-1}) = \lambda\, p_{\mathrm{kNN}}(y_i|x, \hat{y}_{1:i-1}) + (1-\lambda)\, p_{\mathrm{MT}}(y_i|x, \hat{y}_{1:i-1})$$

Final kNN distribution

Decoder's predicted distribution

Khandelwal et al., 2020

Unlike REALM, this approach doesn’t require any training! It retrieves the kNNs via L2 distance using a fast kNN library (FAISS)

This is quite expensive!

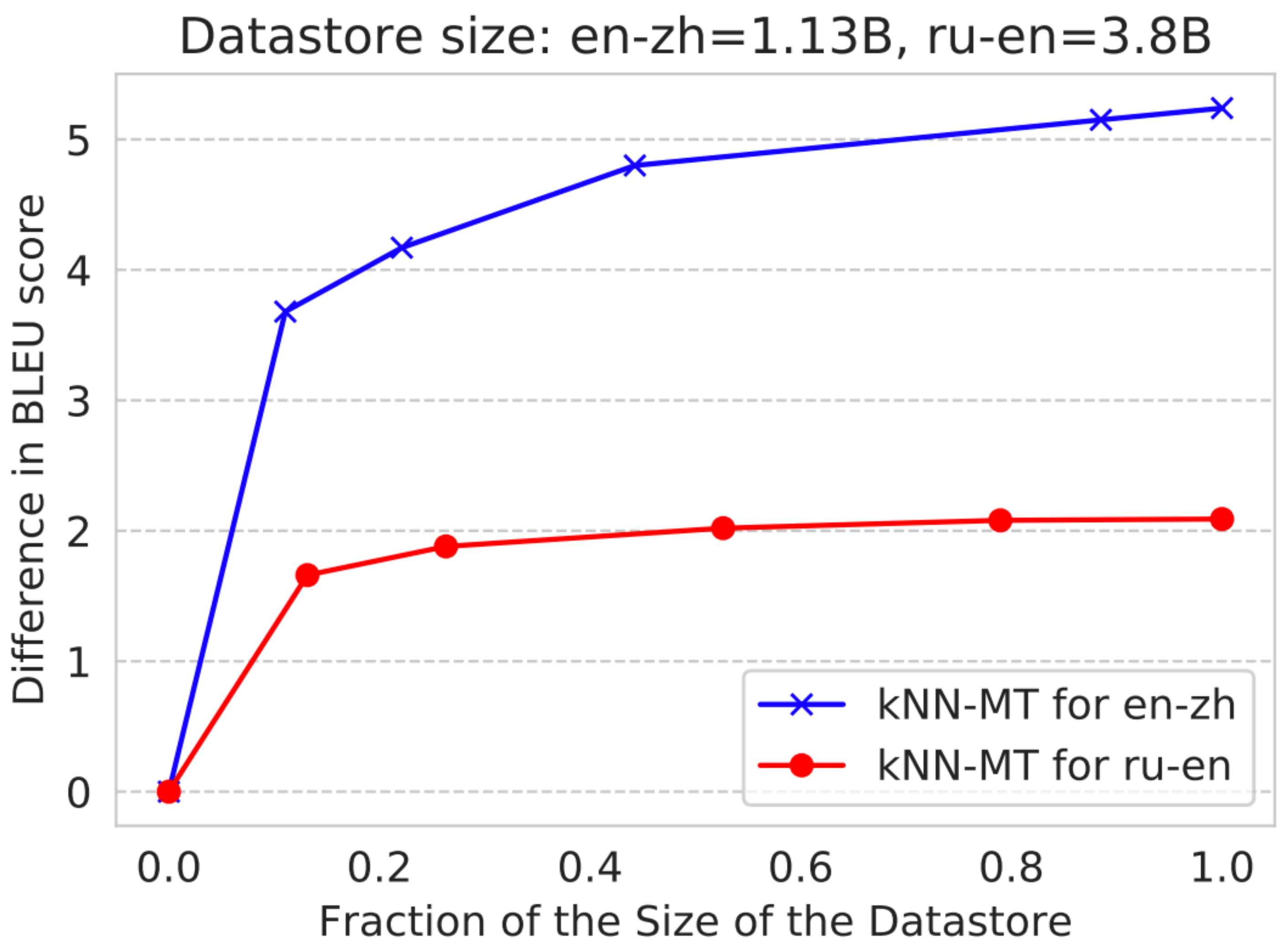
Computational Cost While kNN-MT does not add trainable model parameters, it does add some computational overhead. The primary cost of building the datastore is a single forward pass over all examples in the datastore, which is a fraction of the cost for training on the same examples for one epoch. During inference, retrieving 64 keys from a datastore containing billions of items results in a generation speed that is two orders of magnitude slower than the base MT system.

But also increases translation quality!

	de-en	**ru-en**	**zh-en**	**ja-en**	**fi-en**	**lt-en**	**de-fr**	**de-cs**	**en-cs**
Test set sizes	2,000	2,000	2,000	993	1,996	1,000	1,701	1,997	2,000
Base MT	34.45	36.42	24.23	12.79	25.92	29.59	32.75	21.15	22.78
+kNN-MT	**35.74**	**37.83**	**27.51**	13.14	26.55	29.98	**33.68**	21.62	**23.76**
Datastore Size	5.56B	3.80B	1.19B	360M	318M	168M	4.21B	696M	533M

	en-de	**en-ru**	**en-zh**	**en-ja**	**en-fi**	**en-lt**	**fr-de**	**cs-de**	**Avg.**
Test set sizes	1,997	1,997	1,997	1,000	1,997	998	1,701	1,997	-
Base MT	36.47	26.28	30.22	21.35	21.37	17.41	26.04	22.78	26.00
+kNN-MT	**39.49**	**27.91**	**33.63**	**23.23**	22.20	18.25	**27.81**	23.55	**27.40**
Datastore Size	6.50B	4.23B	1.13B	433M	375M	204M	3.98B	689M	-

Can make it faster by using a smaller datastore



UMass · CS685 | Advanced Natural Language Processing (2020)

CS685 (2020)· 课程资料包 @ShowMeAI

VIDEO PPT

视频 中英双语字幕

课件 一键打包下载

笔记 官方笔记翻译

代码 作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1BL411t7RV

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/umass-cs685

NLP 语言模型 问答系统 文本生成 BERT 语义解析 GPT-3 知识推理 模型蒸馏 迁移学习 transformer 注意力机制

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n

 # Awesome AI Courses Notes Cheatsheets· 持续更新中

知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者？** 回复 [添砖加瓦]