# SIMLM: Pre-training with Representation Bottleneck for Dense Passage Retrieval

Liang Wang, Nan Yang, Xiaolong Huang,
Binxing Jiao Linjun Yang, Daxin Jiang, Rangan Majumder, Furu Wei

ACL 2023

9月28日 ACL読み会

武田・笹野研究室　M2 矢野千紘

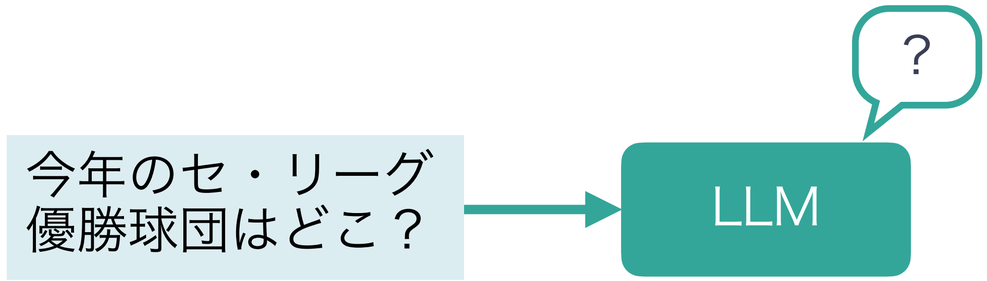
# SimLM

- **密ベクトル検索**のための**事前学習手法**SimLMの提案

# 選んだ理由

- LLMと相性の良い検索モデルについて学びたい
  - LLMを追加学習しなくとも知識を挿入できる
  - Hallucinationを防ぐことができる

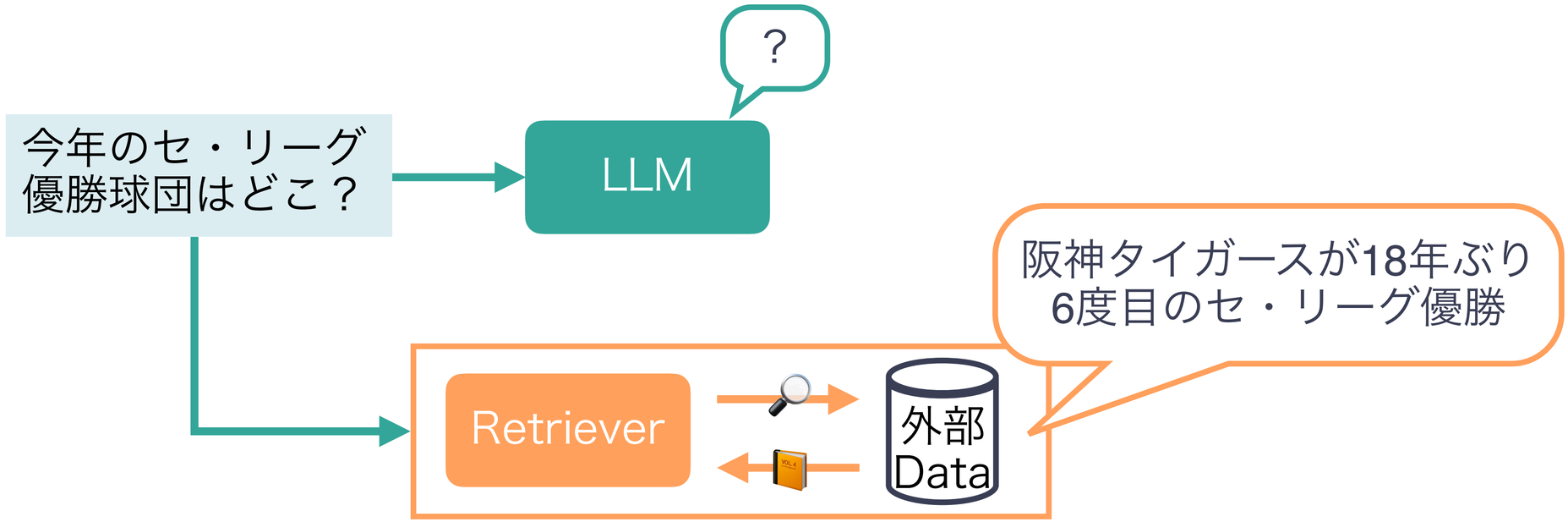
Retrieval Augmented Generation (RAG) の例



# 選んだ理由

- LLMと相性の良い検索モデルについて学びたい
  - LLMを追加学習しなくとも知識を挿入できる
  - Hallucinationを防ぐことができる

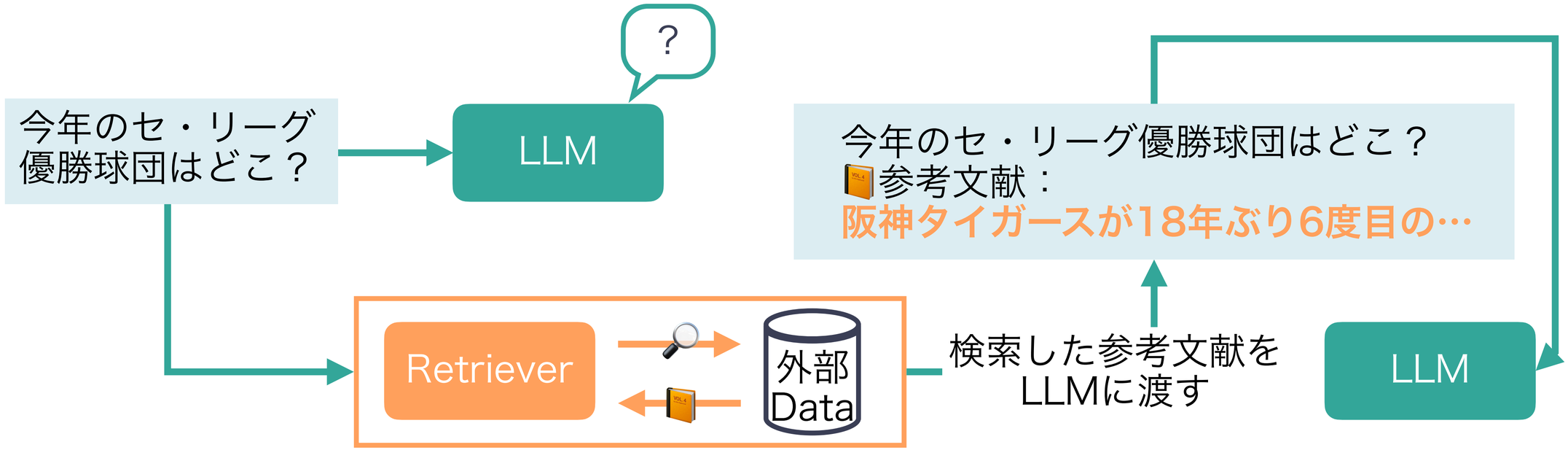
Retrieval Augmented Generation (RAG) の例



# 選んだ理由

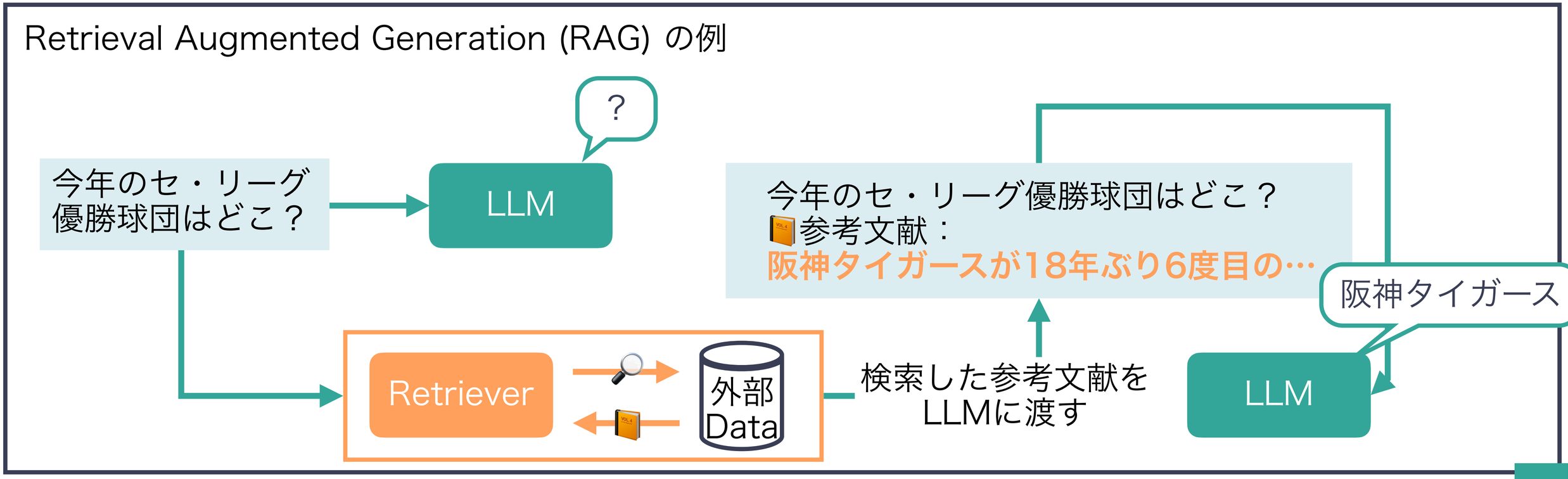
- LLMと相性の良い検索モデルについて学びたい
  - LLMを追加学習しなくとも知識を挿入できる
  - Hallucinationを防ぐことができる

Retrieval Augmented Generation (RAG) の例



# 選んだ理由

- LLMと相性の良い検索モデルについて学びたい
  - LLMを追加学習しなくとも知識を挿入できる
  - Hallucinationを防ぐことができる



# Passage retrieval

- 大量のパッセージ群からクエリに関連するパッセージを検索してくる

✦ **疎ベクトル検索**

- キーワード検索
- (例)：TF-IDF, BM25

✦ **密ベクトル検索**

- ベクトルに変換し、その類似度で検索
- (例)：OpenAIのEmbedding APIのコサイン類似度で検索する

# Passage retrieval

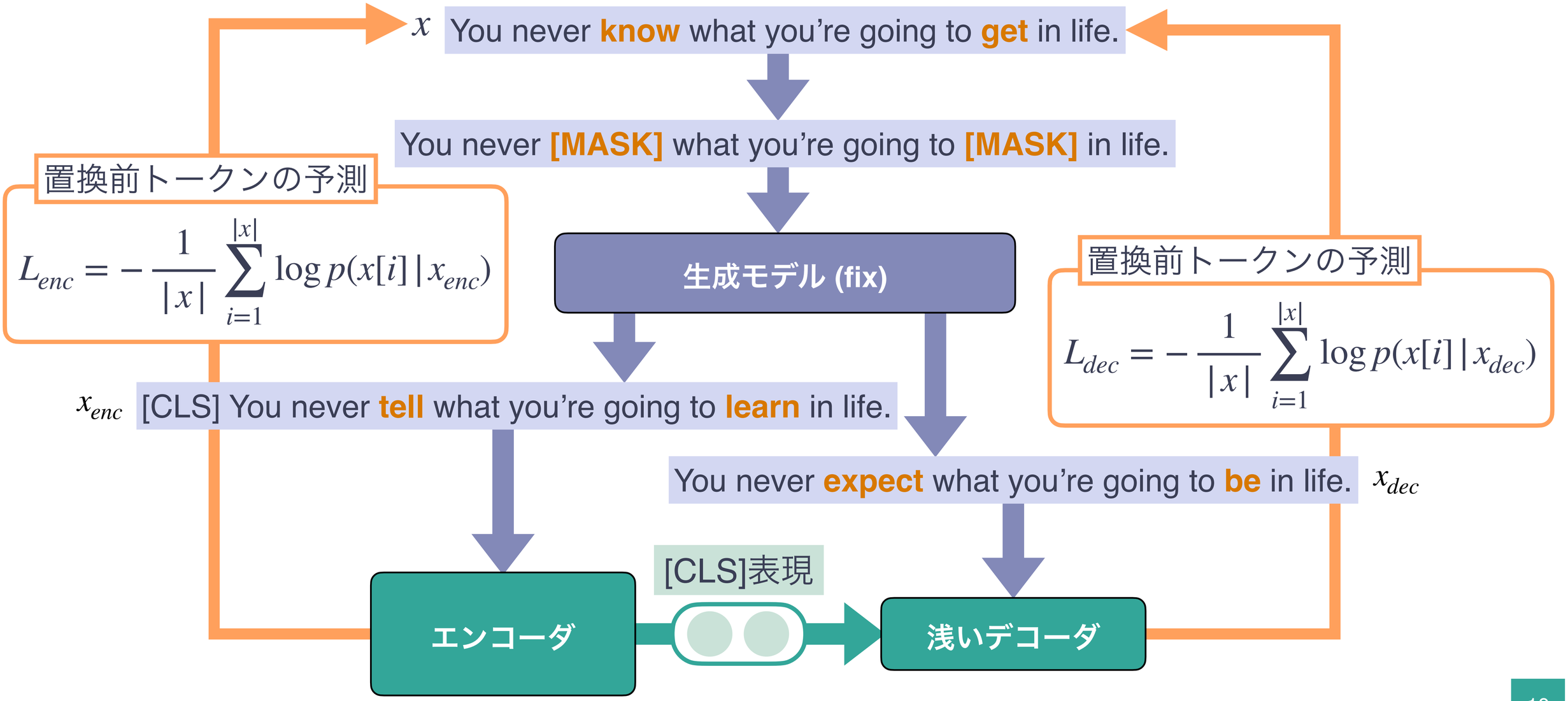
## ✦ 密ベクトル検索

- DPR
  - クエリ用と文章用の2つのエンコーダを利用する (Bi/Dual-encoder)
  - Faissによる近傍探索を行うことでクエリ文章ペアを見つける
- ColBERT
  - クエリ、文章全体ではなく、トークンレベルのベクトルを利用してより高精度な類似度を計算する (multi vector)
  - 保存するべき情報が多く、格納コストが高い（SimLMと比較して6倍程度）
- Condenser:
  - クエリ、文章全体をベクトルとする際の表現ボトルネック ([CLS]など) がより意味情報を圧縮した表現になるように事前学習を行う

# SimLM

- 密ベクトル検索のための**事前学習手法**SimLMの提案
  - エンコーダの出力する**表現ボトルネックが、全ての意味情報を圧縮することよう**に学習させ、良好な性能を示した
- 対照学習と知識蒸留によるFine-tuningを行うことで、ColBERTのようなmulti vectorな手法に勝る性能を示した

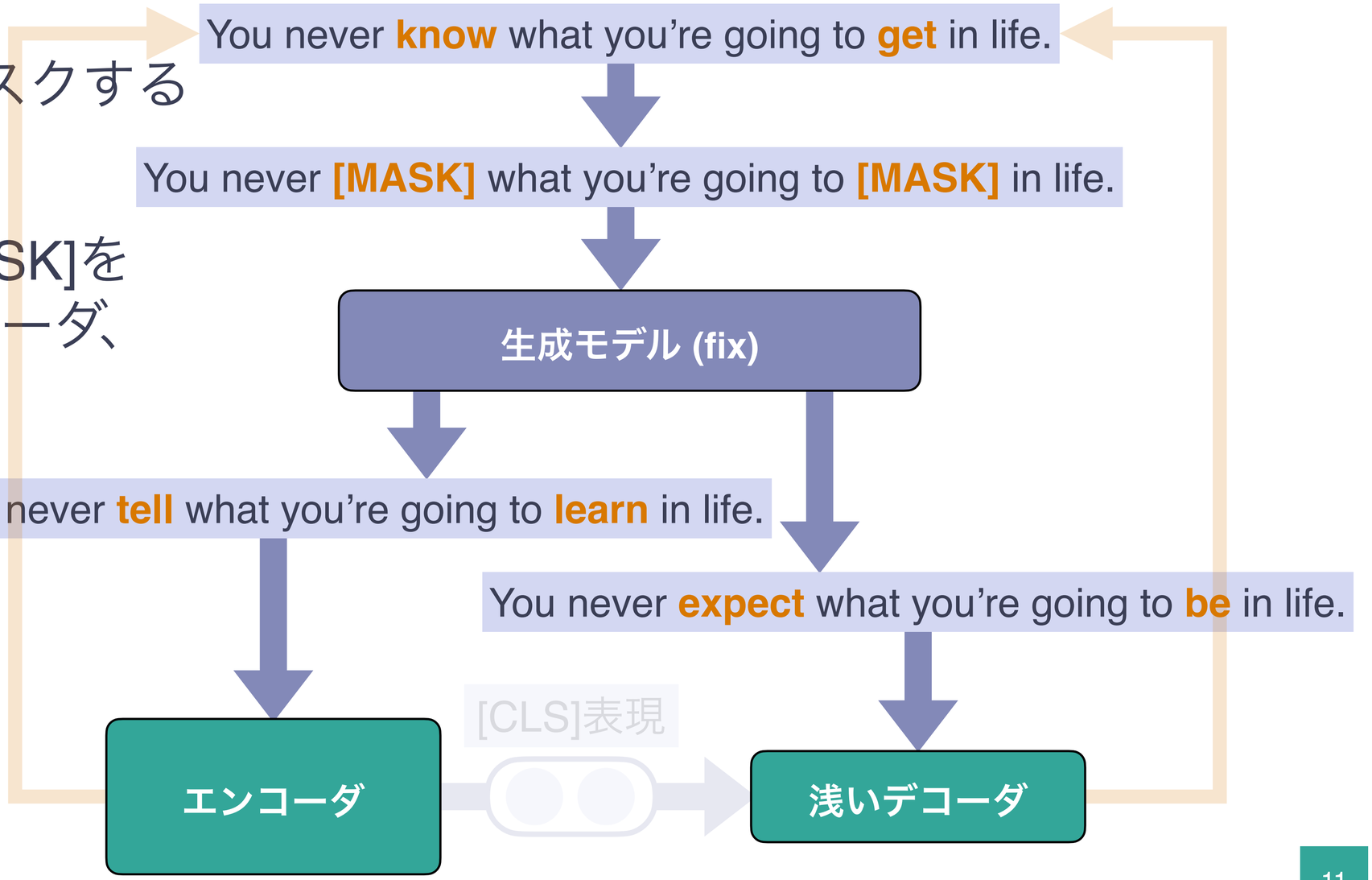
# SimLM：事前学習


$x$
You never know what you're going to get in life.
You never [MASK] what you're going to [MASK] in life.
生成モデル (fix)
置換前トークンの予測
$L_{enc} = -\frac{1}{|x|}\sum_{i=1}^{|x|}\log p(x[i]\,|\,x_{enc})$
$x_{enc}$
[CLS] You never tell what you're going to learn in life.
You never expect what you're going to be in life.
$x_{dec}$
置換前トークンの予測
$L_{dec} = -\frac{1}{|x|}\sum_{i=1}^{|x|}\log p(x[i]\,|\,x_{dec})$
[CLS]表現
エンコーダ
浅いデコーダ

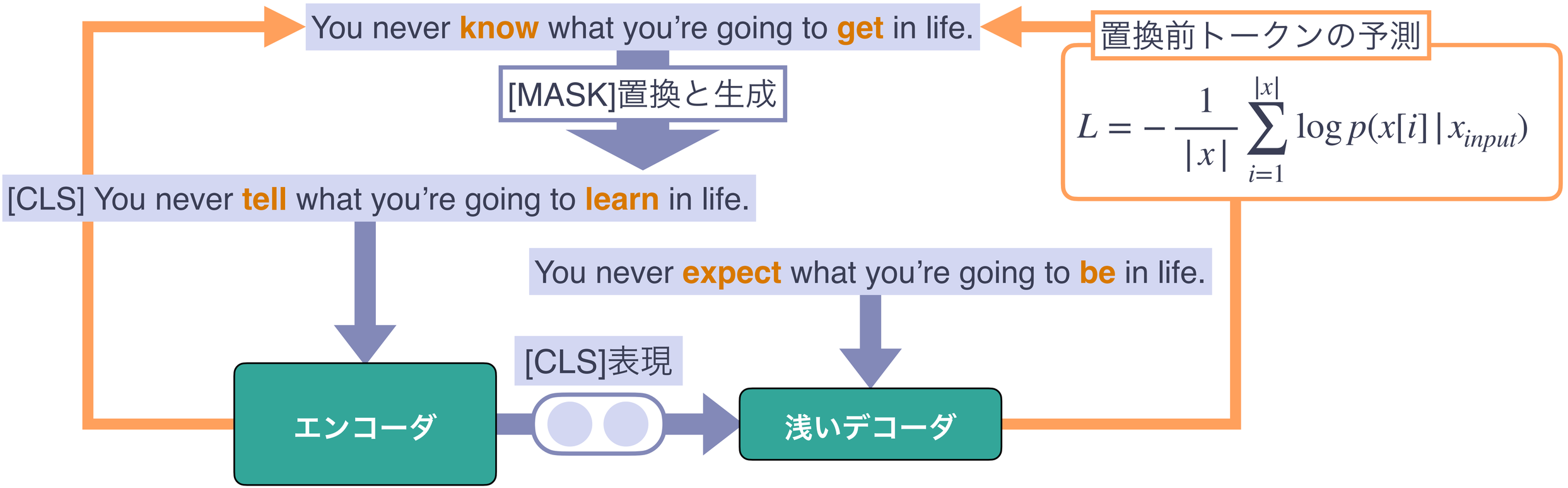
# SimLM：事前学習

1. 確率pで文の一部をマスクする

2. 生成モデルにより[MASK]をそれぞれ置換し、エンコーダ、デコーダへの入力とする



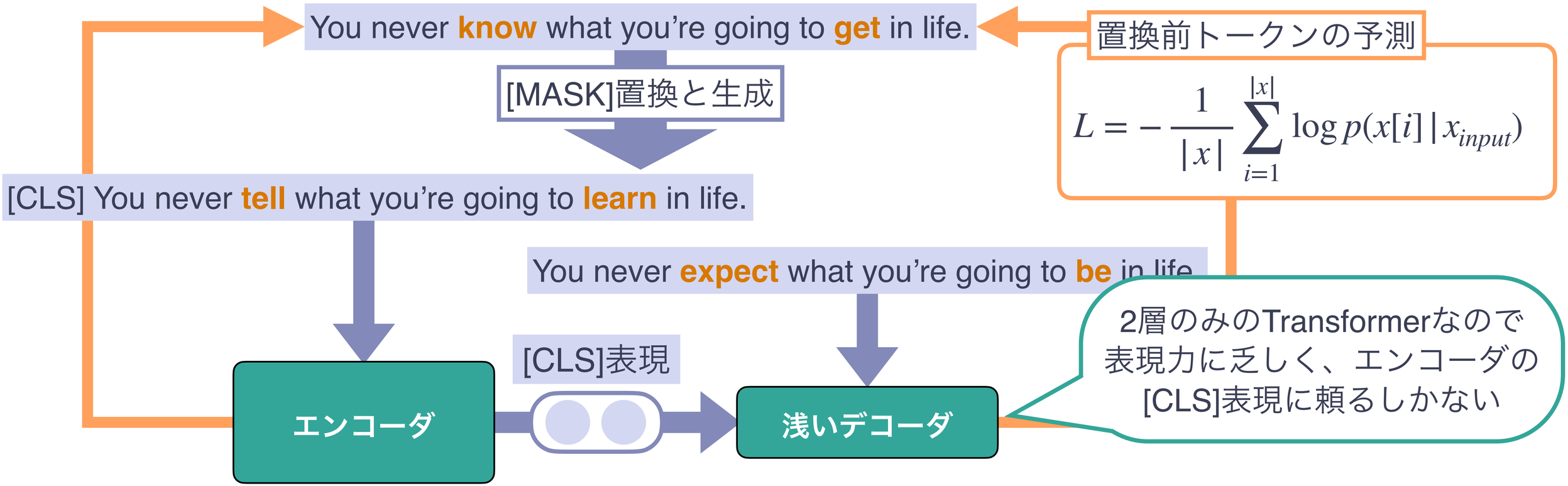
# SimLM：事前学習




3. エンコーダは文全体を受け取り、全てのトークンについて置換前を予測する

4. デコーダはエンコーダの[CLS]表現と文全体を受け取り、全てのトークンについて置換前を予測する

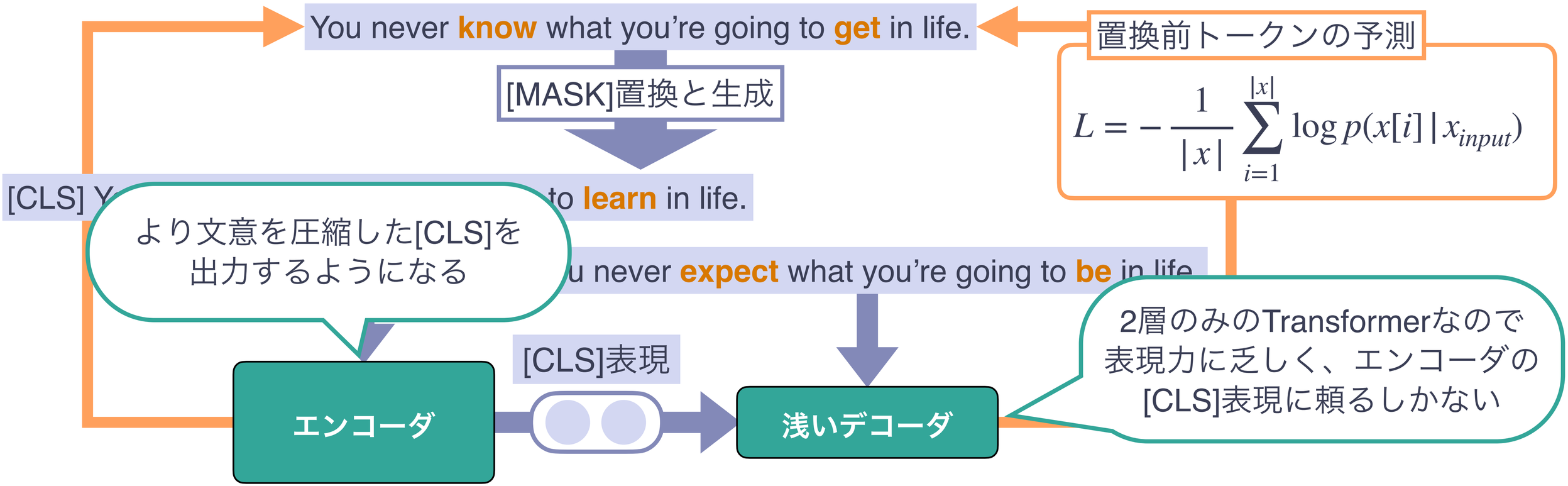
# SimLM：事前学習




3. エンコーダは文全体を受け取り、
全てのトークンについて置換前を
予測する

4. デコーダはエンコーダの
[CLS]表現と文全体を受け取り、
全てのトークンについて置換前を
予測する

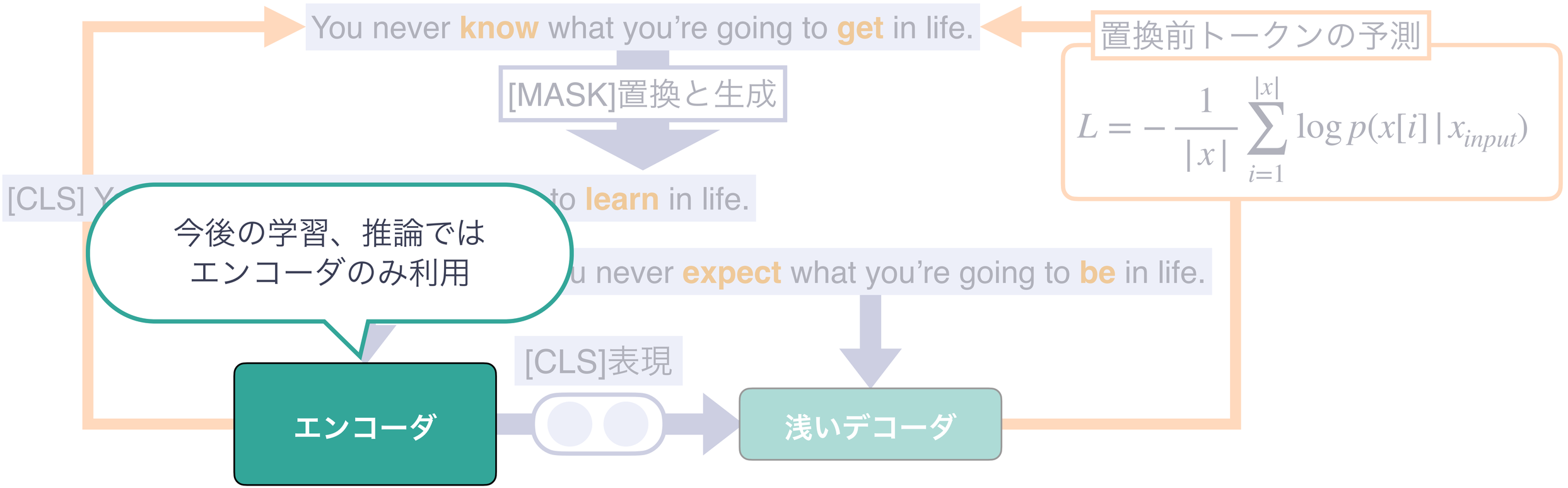
# SimLM：事前学習




3. エンコーダは文全体を受け取り、全てのトークンについて置換前を予測する

4. デコーダはエンコーダの[CLS]表現と文全体を受け取り、全てのトークンについて置換前を予測する

# SimLM：事前学習

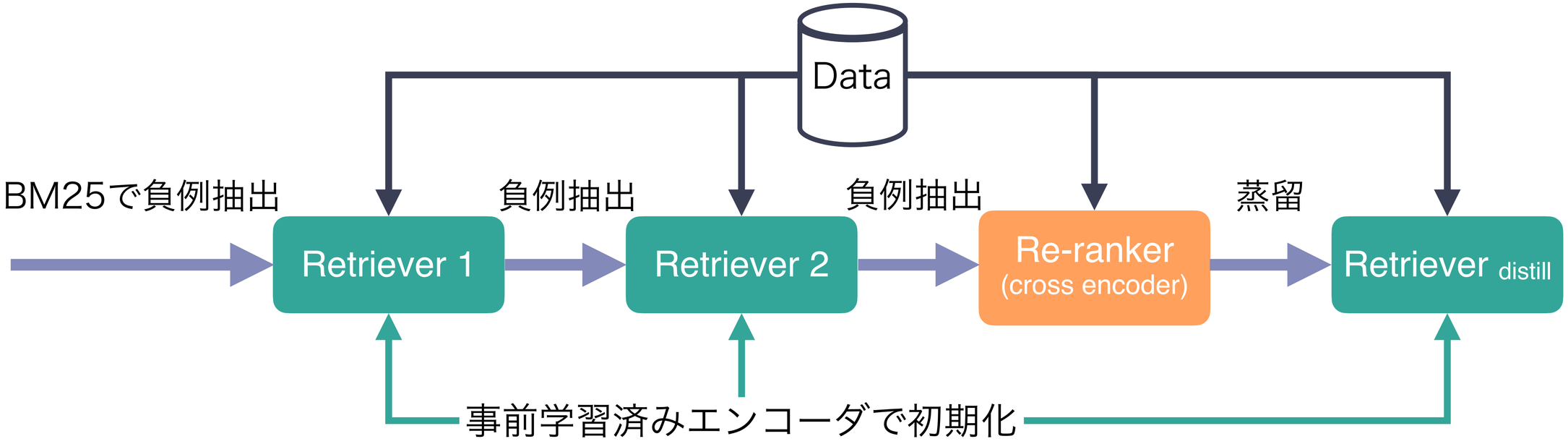



3. エンコーダは文全体を受け取り、全てのトークンについて置換前を予測する

4. デコーダはエンコーダの[CLS]表現と文全体を受け取り、全てのトークンについて置換前を予測する

# SimLM： Fine-tuning

- 同じデータで複数回学習
- 前ステージのモデル出力を学習に利用し、どんどん性能を上げる

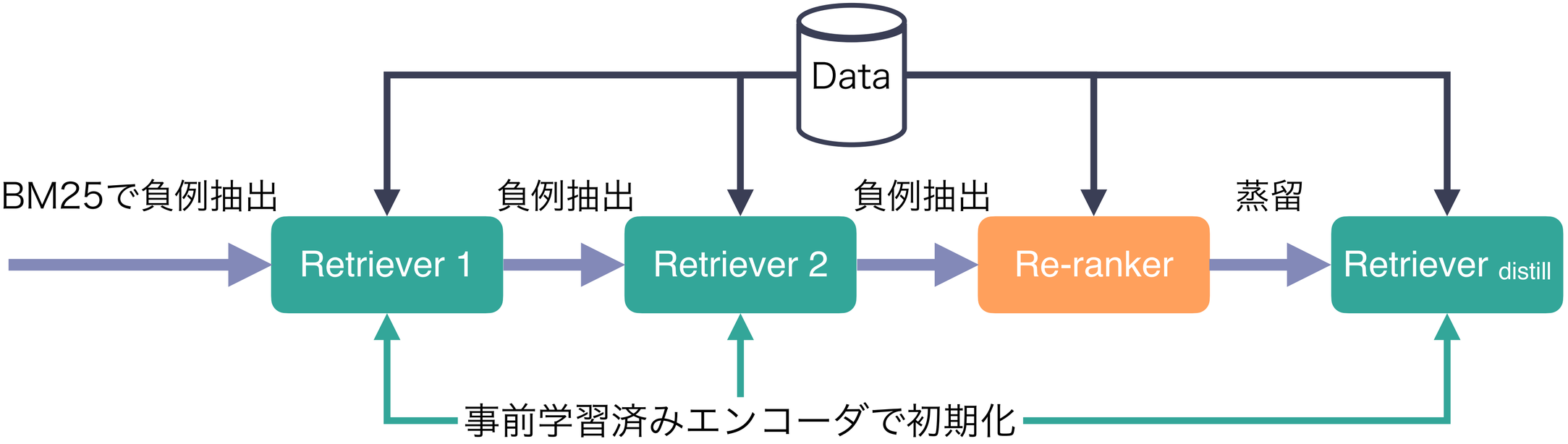


# SimLM： Fine-tuning

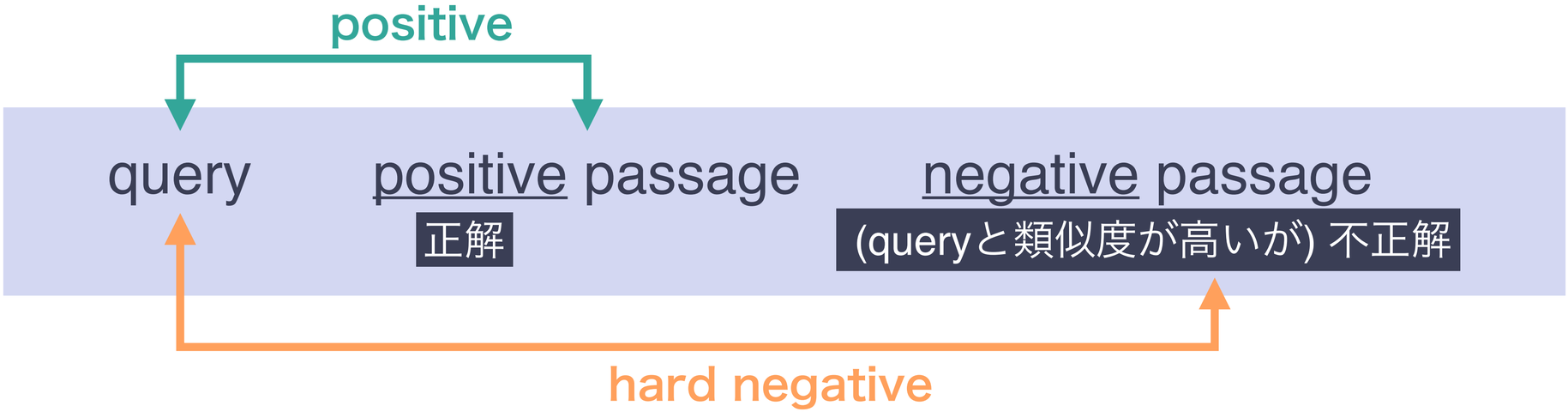
## ✓Retriever1

BM25（疎ベクトル検索の手法）で抽出されたhard negativeを利用し、対照学習を行う

- クエリに対して類似度の高いn件から正解以外のものをhard negativeとする

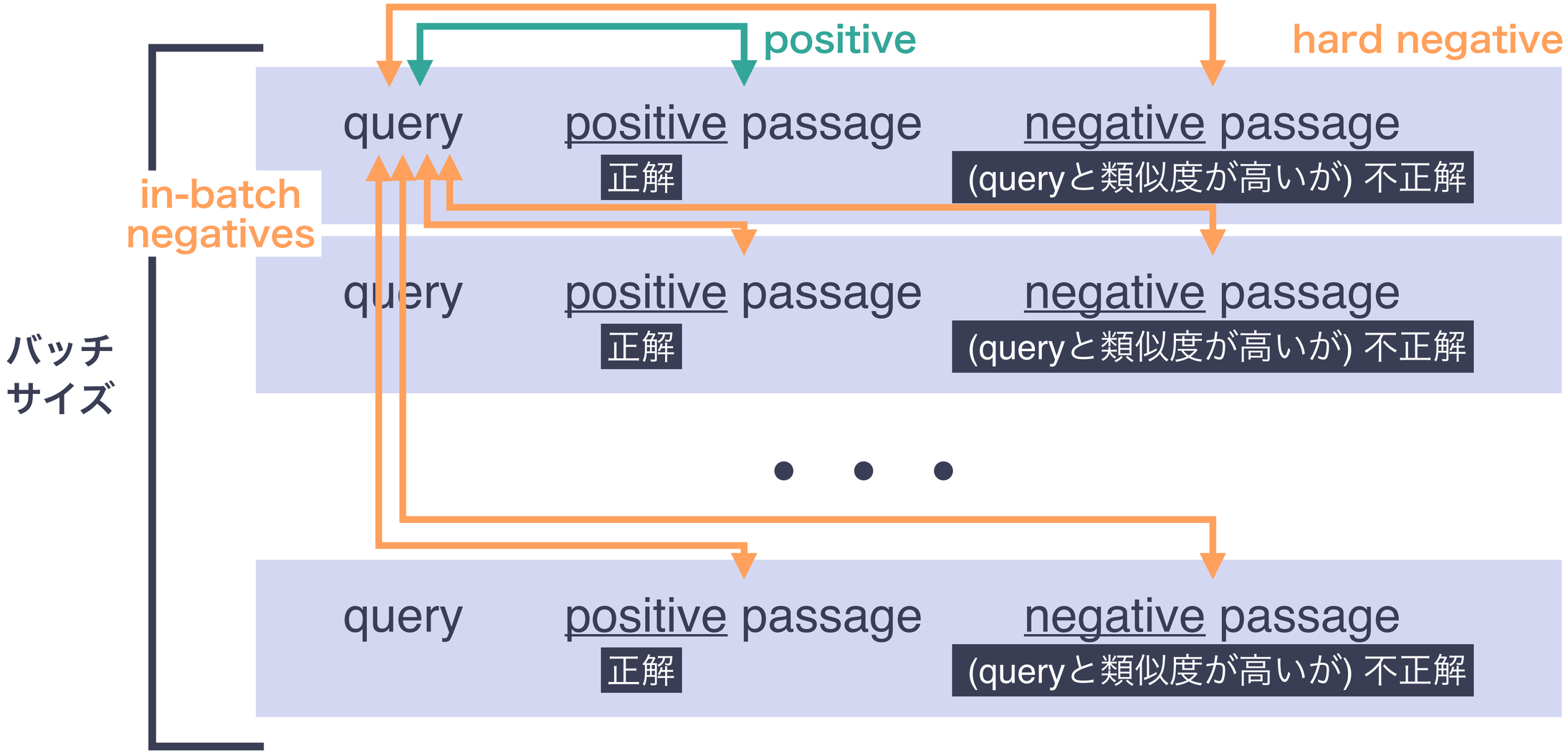


# 対照学習




- positiveペアの類似度が高く、negativeペアの類似度が低くなるように学習

# 対照学習


positive
hard negative
query
positive passage
正解
negative passage
(queryと類似度が高いが) 不正解
in-batch
negatives
query
positive passage
正解
negative passage
(queryと類似度が高いが) 不正解
バッチ
サイズ
query
positive passage
正解
negative passage
(queryと類似度が高いが) 不正解

# 対照学習

- $(q^+, d^+)$: 正解のquery, passageペア
- $\phi$: 類似度関数 コサイン類似度に温度パラメータを導入したもの
- $\mathbb{N}$: 全負例の集合

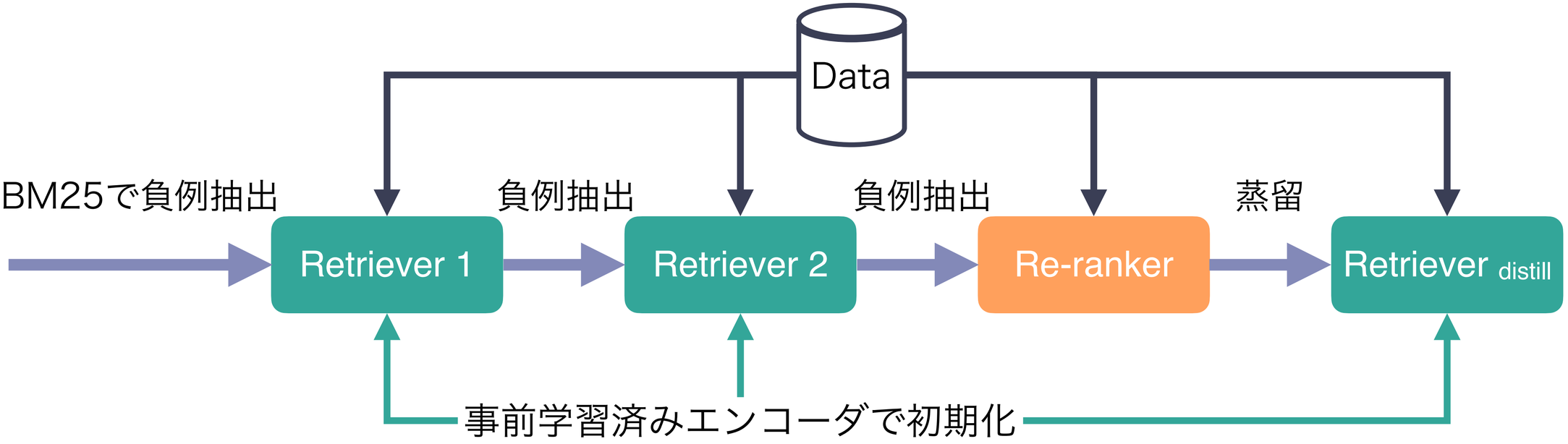
$$L_{cnt} = -\log \frac{\phi(q^+, d^+)}{\phi(q^+, d^+) + \sum_{n_i \in \mathbb{N}} (\phi(q^+, n_i) + \phi(d^+, n_i))}$$

# SimLM： Fine-tuning

## ✓Retriever1

BM25（疎ベクトル検索の手法）で抽出されたhard negativeを利用し、対照学習を行う

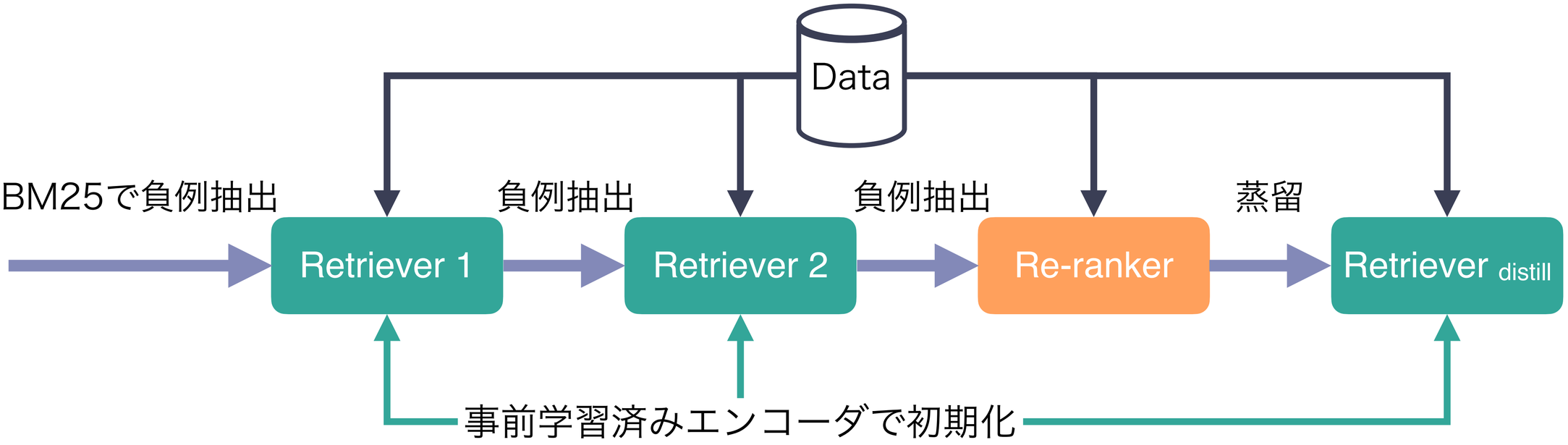
- クエリに対して類似度の高いn件から正解以外のものをhard negativeとする



# SimLM： Fine-tuning

✓**Retriever 2**

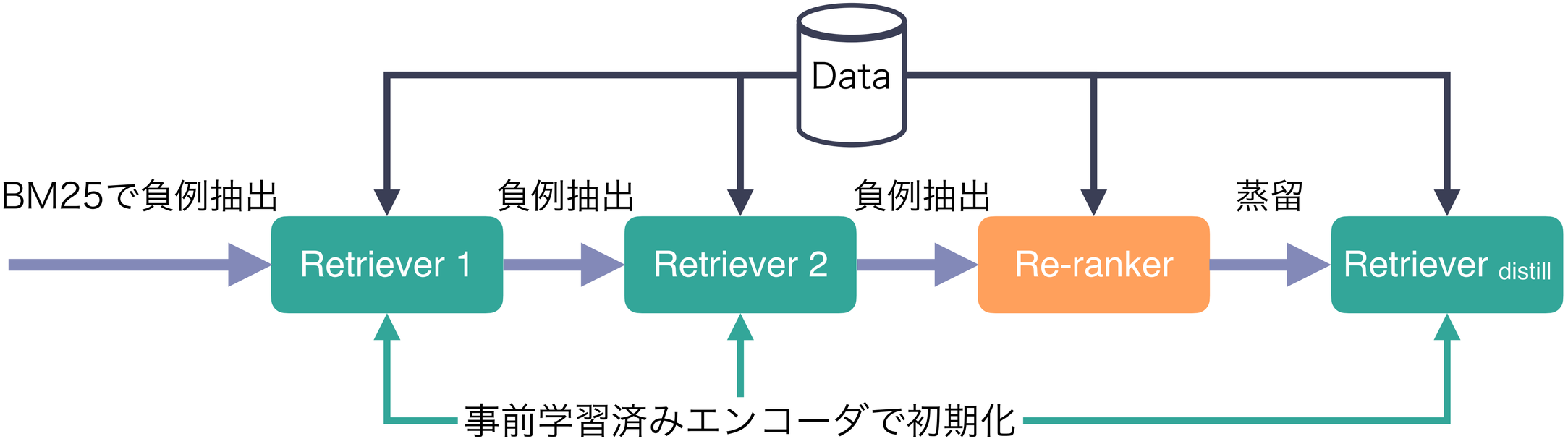
Retriever 1 で抽出されたhard negativeを利用し、対照学習を行う



# SimLM： Fine-tuning

✓Re-ranker

Retriever 2 で抽出されたhard negativeを利用し、正例のスコアが高く、負例のスコアが低くなる様に学習する

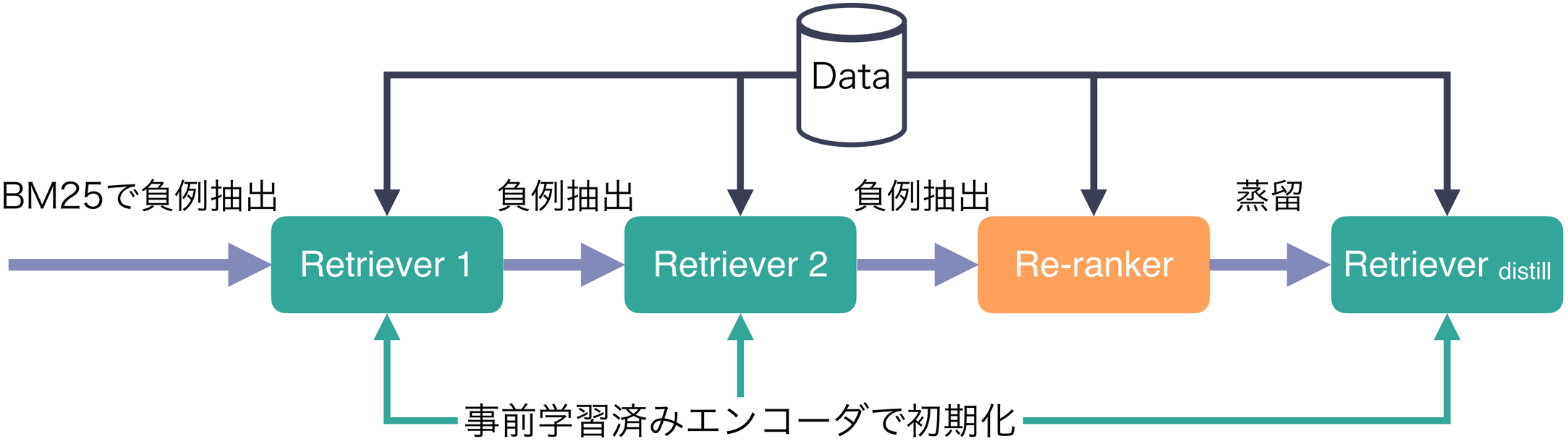


# SimLM： Fine-tuning

✓**Retriever** distill

Re-rankerから知識蒸留

- cross-encoderベースのモデルは強力だが、非実用的
  - 検索するたびに全ての文章をベクトル化しないといけない



# SimLM： Fine-tuning

✓**Retriever distill**

ふたつの損失の線形補完和を損失として利用し学習

- Retriever 2 で抽出されたhard negativeを利用したContrastive Loss
- Re-rankerの付与するスコア分布とのKL divergence
  - Re-rankerとRetrieverの付与するスコア分布が近くなるように学習

$$L_{kl} = \sum_{i=1}^{n} p_{ranker}^{i} log \frac{p_{ranker}^{i}}{p_{ret}^{i}}$$

$$L = L_{cnt} + \alpha L_{kl}$$

# データセット

- MS MARCO
  - とても大規模な質問応答データセット
  - 50万件のラベル付きクエリと880万件の文章から構成
- NQ
  - 8万の質問、回答ペアとWikipedia中の210万件の文章から構成される質問応答データセット

# 評価指標

- Mean reciprocal rank (MRR)
  - 正解した予測についてその順位の逆数の平均

| 予測(スコア降順) | 正解 | 順位 |
|---|---|---|
| dog, cat, **inu** | inu | 3 |
| mikan, **make**, man | make | 2 |
| one, two, ten | zero | - |

(例) MRR@3 $\frac{1}{3} \times (\frac{1}{3} + \frac{1}{2}) = 0.278$

- Recall
  - 予測に正解が含まれる確率

(例) R@3 $\frac{2}{3} = 0.667$

# 性能評価実験

| Model | +distill | single vector? | MS MARCO dev | | |
|---|---|---|---|---|---|
| | | | MRR@10 | R@50 | R@1k |
| **Sparse retrieval** | | | | | |
| BM25 | | ✓ | 18.5 | 58.5 | 85.7 |
| DeepCT (Dai and Callan, 2019) | | ✓ | 24.3 | 69.0 | 91.0 |
| docT5query (Nogueira and Lin) | | ✓ | 27.7 | 75.6 | 94.7 |
| **Dense retrieval** | | | | | |
| ⋮ | | | | | |
| coCondenser (Gao and Callan, 2022) | | ✓ | 38.2 | 86.5* | 98.4 |
| RocketQAv2 (Ren et al., 2021b) | ✓ | ✓ | 38.8 | 86.2 | 98.1 |
| AR2 (Zhang et al., 2021) | ✓ | ✓ | 39.5 | **87.8** | 98.6 |
| ColBERTv2 (Santhanam et al., 2021) | ✓ | | 39.7 | 86.8 | 98.4 |
| SIMLM | ✓ | ✓ | **41.1** | **87.8** | **98.7** |

# 性能評価実験

- multi-vector (文章ではなくトークンごとにひとつのベクトルを格納する)
  手法であるColBERTを凌駕する性能

| Model | +distill | single vector? | MS MARCO dev | | |
|---|---|---|---|---|---|
| | | | MRR@10 | R@50 | R@1k |
| **Sparse retrieval** | | | | | |
| BM25 | | ✓ | 18.5 | 58.5 | 85.7 |
| DeepCT (Dai and Callan, 2019) | | ✓ | 24.3 | 69.0 | 91.0 |
| docT5query (Nogueira and Lin) | | ✓ | 27.7 | 75.6 | 94.7 |
| **Dense retrieval** | | | | | |
| ⋮ | | | | | |
| coCondenser (Gao and Callan, 2022) | | ✓ | 38.2 | 86.5* | 98.4 |
| RocketQAv2 (Ren et al., 2021b) | ✓ | ✓ | 38.8 | 86.2 | 98.1 |
| AR2 (Zhang et al., 2021) | ✓ | ✓ | 39.5 | **87.8** | 98.6 |
| ColBERTv2 (Santhanam et al., 2021) | ✓ | | 39.7 | 86.8 | 98.4 |
| SIMLM | ✓ | ✓ | **41.1** | **87.8** | **98.7** |

# 学習パイプラインの影響分析

MS-MARCOで評価

| | MRR@10 | R@1k |
|---|---|---|
| **coCondenser** | | |
| BM25 negatives | 35.7 | 97.8 |
| + mined negatives | 38.2 | 98.4 |
| + distillation | 40.2* | 98.3* |
| **SIMLM** | | |
| BM25 negatives ($\text{Retriever}_1$) | 38.0 | 98.3 |
| + mined negatives ($\text{Retriever}_2$) | 39.1 | 98.6 |
| + distillation ($\text{Retriever}_{\text{distill}}$) | **41.1** | **98.7** |
| Cross-encoder re-ranker | 43.7 | 98.6 |

- 負例の抽出やRe-rankerの蒸留で性能向上している
- 負例を抽出する段階を増やしても有意な性能の変化はなかった

# 事前学習時の目的関数について

MS-MARCOで評価

| | SIMLM | Enc-Dec MLM | Condenser | MLM | Enc-Dec RTD | AutoEncoder | $\text{BERT}_{\text{base}}$ |
|---|---|---|---|---|---|---|---|
| MRR@10 | **38.0** | 37.7 | 36.9 | 36.7 | 36.2 | 32.8 | 33.7 |

- Enc-Dec MLM: 同じアーキテクチャで生成器を使用せず、[MASK]部分の予測を行う
- Condenser: 表現ボトルネックに文意を圧縮するような事前学習アーキテクチャ (浅い層の文全体の表現と深い層の[CLS]表現を使ってMLMで学習)
- MLM: エンコーダのみ[MASK]部分の予測を行う
- Enc-Dec RTD: 同じアーキテクチャで置換トークンの検出を行う
- AutoEncoder: エンコーダの[CLS]からデコーダが入力を復元する

# 事前学習時の目的関数について

MS-MARCOで評価

| | SIMLM | Enc-Dec MLM | Condenser | MLM | Enc-Dec RTD | AutoEncoder | $BERT_{base}$ |
|---|---|---|---|---|---|---|---|
| MRR@10 | **38.0** | 37.7 | 36.9 | 36.7 | 36.2 | 32.8 | 33.7 |

- 同じアーキテクチャで比較すると、SimLM > Enc-Dec MLM >> Enc-Dec RTD
- 検索タスクにおいては MLM > RTD である
  - (私信) SimLMの目的関数は置換前の予測であり、MLMと性能が近いことから識別するようなタスクよりトークンを予測するようなタスクの方が良い？

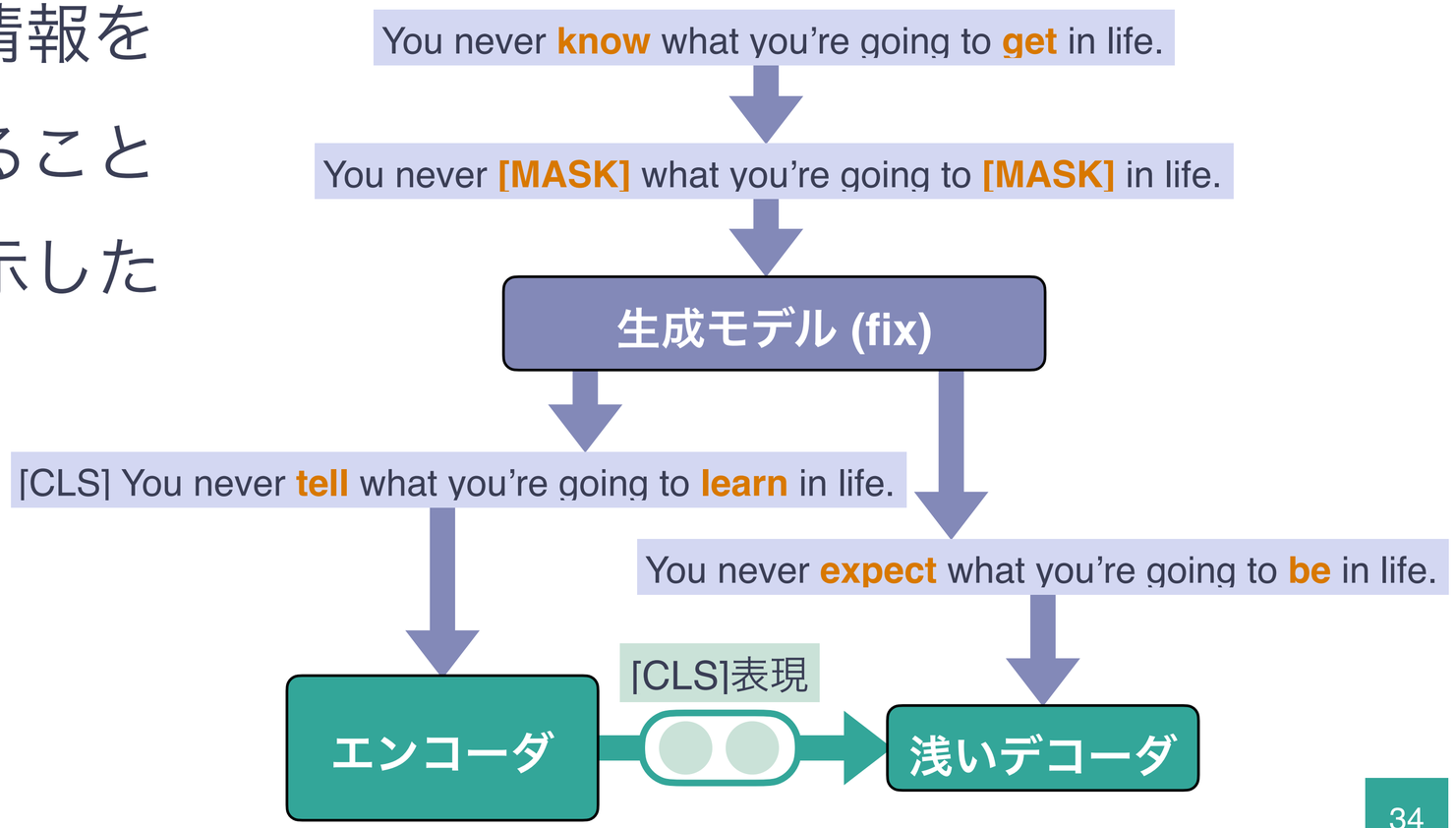
# 事前学習コーパスによる影響

| Corpus | MS-MARCO | | NQ | |
|---|---|---|---|---|
| | MRR@10 | R@1k | R@20 | R@100 |
| none | 33.7 | 95.9 | 82.9 | 88.0 |
| MS-MARCO | **38.0** | **98.3** | 83.3 | 88.6 |
| Wikipedia | 36.3 | 97.4 | **84.3** | **89.3** |

- MS-MARCOではBM25による負例、NQでは抽出された負例を利用したFine-tuningを行ってから評価
- 事前学習とFine-tuning、評価データが同じ時高い性能
- 噛み合っていない時でも性能の向上は大きく、事前学習によって情報を[CLS]表現に圧縮する能力が獲得されている

# まとめ

- 密ベクトル検索のための事前学習手法SimLMの提案
- エンコーダの出力のうち、表現ボトルネック (今回は[CLS]) を浅いデコーダに渡すことで、全ての意味情報を表現ボトルネックに圧縮することを学習させ、良好な性能を示した



# 感想

- とてもコストがかかって大変そう
  - 対象データごとに事前学習が必要なのは辛い
- 表現ボトルネックを良くするための事前学習アーキテクチャが面白い
  - 検索界隈には色々ありそう？
  - Condenserなど…