NEURAL DOCUMENT RETRIEVAL

Martin Fajčík

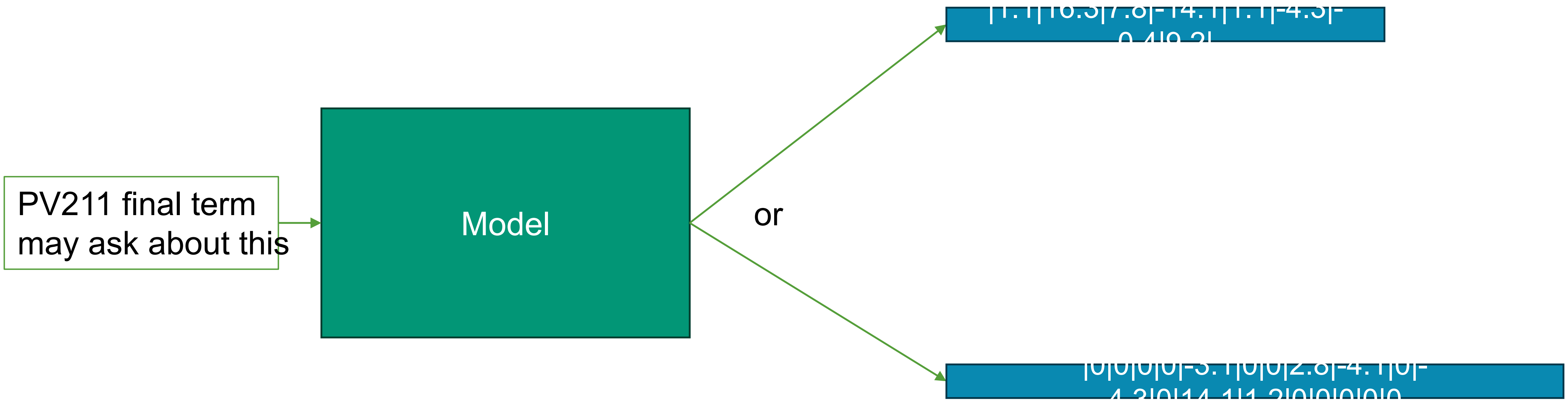
MUNI PV211 2024

Last edited: 15.04.2024

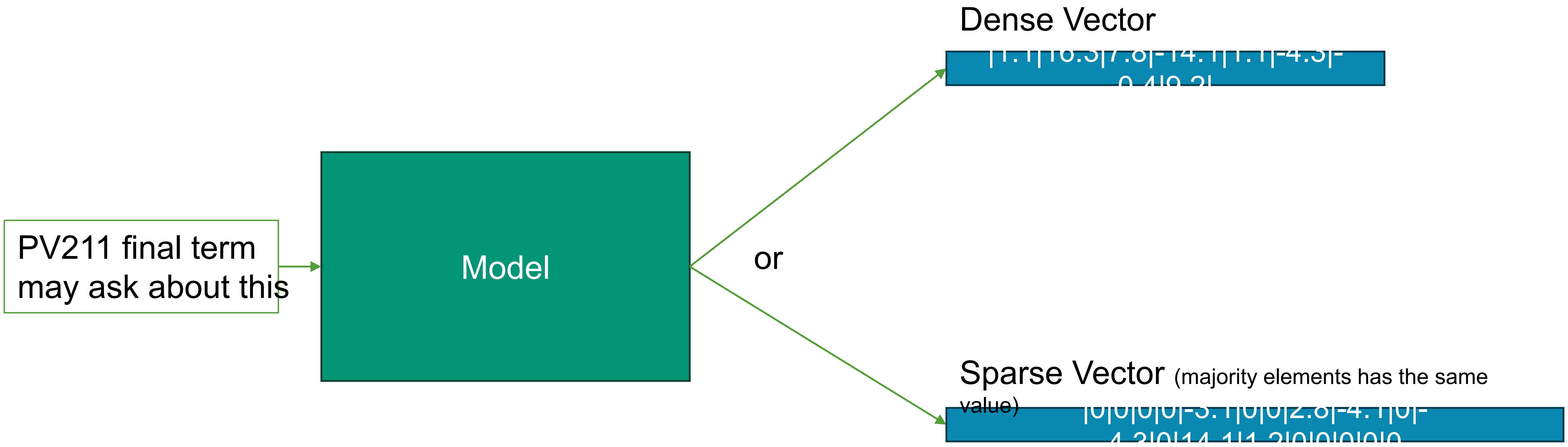
Outline

- Dense retrieval vs Sparse retrieval
- Dense Retrieval – Dense Passage Encoder
- Neural Reranking – Cross-Encoder
- Dense Retrieval – Contriever
- Sparse Retrieval – SPLADE
- Dense Retrieval – ColBERT

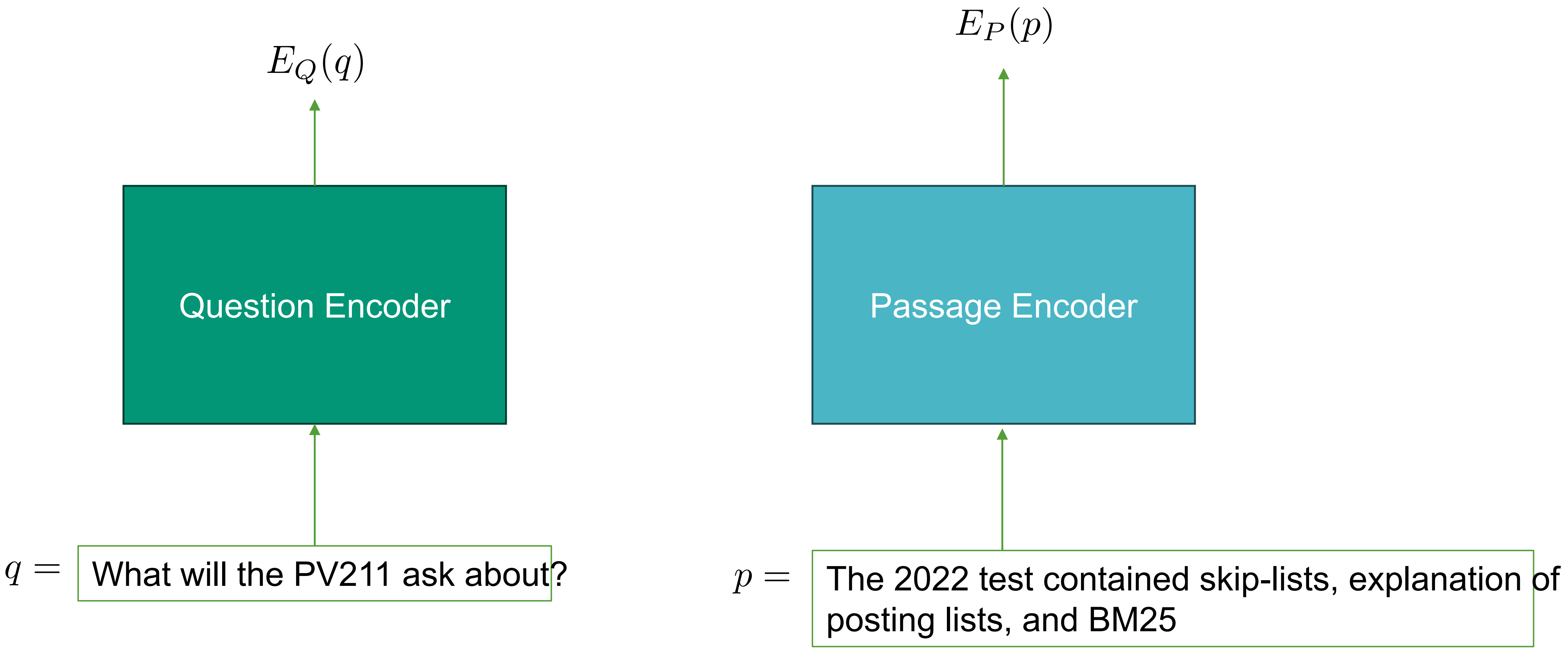
Dense Retrieval vs Sparse Retrieval



Dense Retrieval vs Sparse Retrieval



Dense Passage Retrieval



Karpukhin, Vladimir, et al. "Dense Passage Retrieval for Open-Domain Question Answering." *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020.

Dense Passage Retrieval

- The similarity function between two representations

$$\mathrm{sim}(q, p) = E_Q(q)^\top E_P(p)$$

- Assume dataset with *m* training instances

$$\mathcal{D} = \{(q_i, p_i^+, p_{i,1}^-, p_{i,2}^-, ..., p_{i,n}^-)\}_{i=1}^m$$

- Each instance contains one question q_i and one relevant (positive) passage p_i^+ along with *n* irrelevant (negative) passages $p_{i,j}^-$

Karpukhin, Vladimir, et al. "Dense Passage Retrieval for Open-Domain Question Answering." *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020.

Dense Passage Retrieval Training

- Cross-Entropy Contrastive Loss

$$L(q_i, p_i^+, p_{i,1}^-, p_{i,2}^-, ..., p_{i,n}^-) = -\log \frac{\exp(\mathrm{sim}(q_i, p_i^+))}{\exp(\mathrm{sim}(q_i, p_i^+)) + \sum_{j=1}^{n} \exp(\mathrm{sim}(q_i, p_{i,j}^-))}$$

- Hyperparameters and Model choices
  - Encoder and Decoder are both based on **BERT** pre-trained models
  - 21M of **100 words** passages of Wikipedia were indexed
  - Each passage is also prepended with the **title** of the Wikipedia article where the passage is from, separated with a [SEP] token
  - batch size **128**
  - Learning rate 1e-5, using Adam optimizer, linear scheduling with warmup rate 0.1, and dropout 0.1
  - Training sets of each dataset had around **~60k** samples

Karpukhin, Vladimir, et al. "Dense Passage Retrieval for Open-Domain Question Answering." *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020.

Dense Passage Retrieval Inference

- Index Construction
  - Create embeddings for all (21M) passages, using **passage encoder**.
  - This takes a lot of memory (fp16 of 21M 768 dimensional embeddings ~ 32 GB)

- Query Time
  - Use **query encoder** to encode question.
  - Find **nearest neighbor** doing full dot-product (O(n)) with 21M embeddings.
  - Then compute **arg top-K**, to find K nearest values.
  - (Optional) use ***approximate nearest neighbor methods***, with logarithmic expected computational complexity, such as HNSW.

Karpukhin, Vladimir, et al. "Dense Passage Retrieval for Open-Domain Question Answering." *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020.

Dense Passage Retrieval Negative Sample Mining

- Top BM25 passage that does not contain answer string (original DPR was made on open-domain QA)

- In-batch negatives

- Pros and Cons of both? (discussion)

Karpukhin, Vladimir, et al. "Dense Passage Retrieval for Open-Domain Question Answering." *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020.

Dense Passage Retrieval Negative Sample Mining

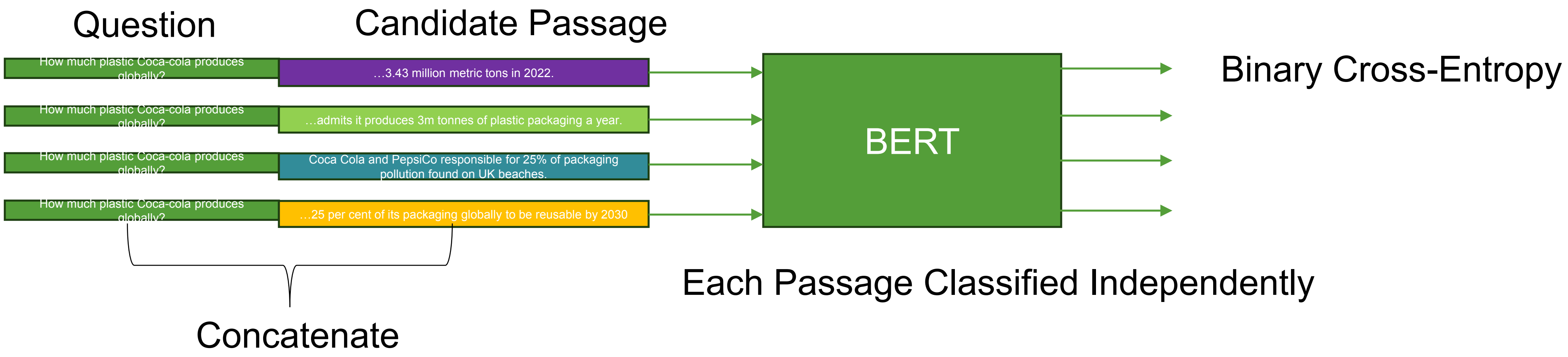
- Top BM25 passage that does not contain answer string (original DPR was made on open-domain QA)
  - C: Answer list is non-exhaustive, possibility of **False negatives**
  - C: Unclear how to mine for non-QA applications
  - P: BM25 is unsupervised
  - P: BM25 provides near-to-relevant negatives

- In-batch negatives
  - P: Cheaply obtained, no need for extra encoding
  - C: Requires **large** batch size

Karpukhin, Vladimir, et al. "Dense Passage Retrieval for Open-Domain Question Answering." *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020.

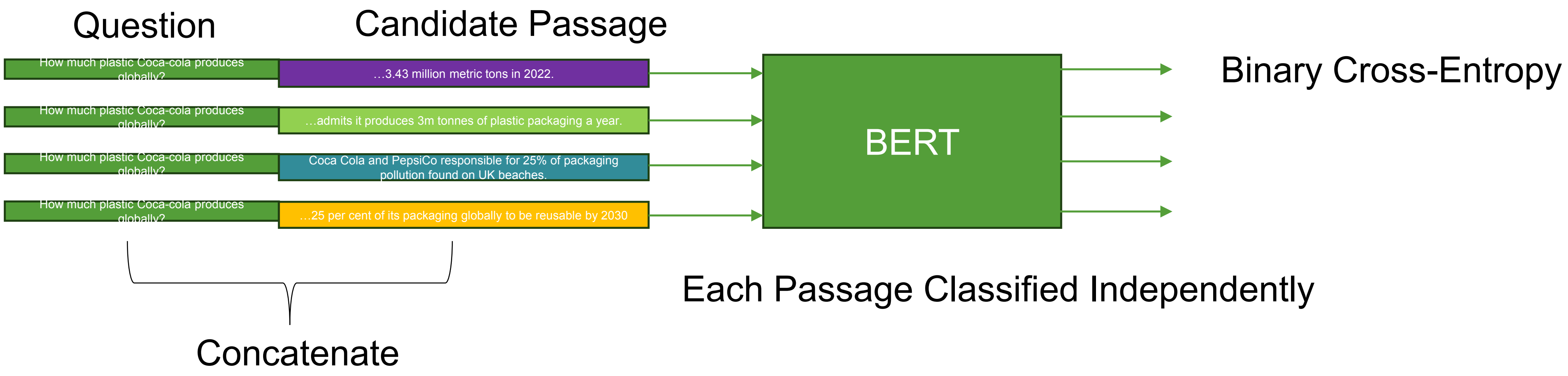
Cross-Encoder Reranking

- Cross-Encoder tends to work better then Bi-Encoder
- Cross-Encoder can't be used for (First stage) Retrieval



Nogueira, Rodrigo, and Kyunghyun Cho. "Passage Re-ranking with BERT." *arXiv preprint arXiv:1901.04085* (2019).

Cross-Encoder Reranking



- Reranking dataset (from the original paper)
    - Positives: relevant passage from dataset.
    - Negatives: top-1000 non-relevant passages.

Nogueira, Rodrigo, and Kyunghyun Cho. "Passage Re-ranking with BERT." *arXiv preprint arXiv:1901.04085* (2019).

Contriever Parallels with DPR

- Cross-Entropy Loss similar to DPR
  - τ is a temperature hyperparameter (0.05, in pretraining and finetuning)

$$\mathcal{L}(q, k_+) = -\frac{\exp(s(q, k_+)/\tau)}{\exp(s(q, k_+)/\tau) + \sum_{i=1}^{K} \exp(s(q, k_i)/\tau)},$$

- Based on **bi-encoder** BERT architecture (same as DPR), but encoder is **shared**

- Inference is the same as with DPR

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. TMLR.

Contriever Unsupervised Pretraining

- Pretraining with **Independent cropping**
  - **Positive** samples are random subsequences of the same document in pre-training corpus.

  - “We use the random cropping data augmentation, with documents of **256 tokens** and span sizes sampled between **5% and 50% of the document** length. Documents are simply random piece of text sampled from a mix between Wikipedia and CCNet data, where half the batches are sampled from each source. We also apply **token deletion** with a probability of 10%.”

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. TMLR.

Contriever Unsupervised Pretraining

- Pretraining with **Independent cropping**
  - **Negative** samples come from
    - In-batch negatives (with very large batch 2048)
    - MoCo algorithm (He et al., 2020)

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. TMLR.
Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. **Momentum contrast** for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9729–9738, 2020

Contriever Unsupervised Pretraining

- MoCo algorithm
  - Store **representations from last N previous batches** in a queue and use them as (cheap) negative examples in the loss.
  - The gradients for similarity with these representations is only computed w.r.t. parameters of the **“query”** encoder.
  - As the model might sometimes change rapidly, reusing the old representations might **lead to a drop of performance when the network rapidly changes during training.**
    - Hence authors define document encoder is an exponential average of query encoder.
    - At every training update step:
      - Query network θ_q is updated via gradient computed from SGD-like optimizer.
      - Document network θ_d is updated from the new parameters of query network θ_q.

$$\theta_d = m\theta_d + (1 - m)\theta_q$$

      - Authors use *m*=0.9995

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. TMLR.
Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. **Momentum contrast** for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9729–9738, 2020

Contriever Unsupervised Pretraining Hyperparameters

Optimizer: AdamW
Learning rate: 5e-5
Batch size: 2048
Steps: 500,000
Queue: 131,072 (representations from last 64 batches were considered)

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. TMLR.
Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. **Momentum contrast** for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9729–9738, 2020

Contriever Supervised Finetuning

- Batch size 1024
- Learning rate 1e-5
- Negatives:
  - First phase: 20k steps only with in-batch negatives
  - Second phase: in-batch negatives in 90% of cases, 10% of cases are **hard negatives** mined from the model trained in phase 1 (how many is not documented)

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. TMLR.

SPLADE (V2)

- Learning **sparse** vectors for retrieval using BERT-based model.

Formal, Thibault et al. “SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval.” *ArXiv* abs/2109.10086 (2021)

SPLADE (V2)

- Learning **sparse** vectors for retrieval using BERT-based model.

- A sequence of tokens from volabulary **V** set for question / passage:

$$t = (t_1, t_2, ..., t_N)$$

- A sequence of contextualized representations extracted as

$$h_1, h_2, ..., h_N = \mathrm{BERT}(t)$$

Formal, Thibault et al. "SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval." *ArXiv* abs/2109.10086 (2021)

SPLADE (V2)

- A sequence of contextualized representations extracted as

$$h_1, h_2, ..., h_N = \mathrm{BERT}(t)$$

- Next, let's reuse BERT-pretraining "head" to compute importance scores of
  - i-th position representation to the
  - j-th vocabulary token

$$w_{ij} = \mathrm{transform}(h_i)^\top E_j + b_j$$

Formal, Thibault et al. "SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval." *ArXiv* abs/2109.10086 (2021)

SPLADE (V2)

- Next, let's reuse BERT-pretraining "head" to compute importance scores of
  - i-th position representation to the
  - j-th vocabulary token

$$w_{ij} = \text{transform}(h_i)^\top E_j + b_j$$

- **E_j** is a token-embedding matrix of BERT
- **b_j** is a token-embedding bias
- **transform** is a non-linear function $\text{transform}(h_i) = \ln(\text{GeLU}(W h_i))$
- both learned during BERT pretraining

Formal, Thibault et al. "SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval." *ArXiv* abs/2109.10086 (2021)

SPLADE (V2)

$$w_{ij} = \text{transform}(h_i)^\top E_j + b_j$$

- $\mathbf{E_j}$ is a token-embedding matrix of BERT
- $\mathbf{b_j}$ is a token-embedding bias
- **transform** is a non-linear function
- both learned during BERT pretraining

- $\mathbf{w_{ij}}$ are actually pre-softmax scores of i-th representation h_i during pretraining

Formal, Thibault et al. "SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval." *ArXiv* abs/2109.10086 (2021)

SPLADE (V2)

$$w_{ij} = \mathrm{transform}(h_i)^{\top} E_j + b_j$$

$$w_j = \max_{i=0}^{N} \log(1 + \mathrm{ReLU}(w_{i,j})) \quad \mathrm{ReLU}(x) = \max(x, 0)$$

Formal, Thibault et al. “SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval.” *ArXiv* abs/2109.10086 (2021)

SPLADE (V2)

$$w_{ij} = \text{transform}(h_i)^{\top} E_j + b_j$$

$$w_j = \max_{i=0}^{N} \log(1 + \text{ReLU}(w_{i,j})) \qquad \text{ReLU}(x) = \max(x, 0)$$

- Sanity check: What is the size of vector w?

Formal, Thibault et al. "SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval." *ArXiv* abs/2109.10086 (2021)

SPLADE (V2)

$$w_{ij} = \text{transform}(h_i)^T E_j + b_j$$

$$w_j = \sum_{i=0}^{N} \log(1 + \text{ReLU}(w_{i,j})) \qquad \text{ReLU}(x) = \max(x, 0)$$

- Sanity check: What is the size of vector w?
- |w| = |V| (same as size of vocabulary)

Formal, Thibault et al. "SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval." *ArXiv* abs/2109.10086 (2021)

SPLADE Training

- Uses the same style contrastive loss as DPR

$$\mathrm{sim}(q,p) = w_q^\top w_p$$

$$L_{rank}(q_i, p_i^+, p_{i,1}^-, p_{i,2}^-, ..., p_{i,n}^-) = -\log \frac{\exp(\mathrm{sim}(q_i, p_i^+))}{\exp(\mathrm{sim}(q_i, p_i^+)) + \sum_{j=1}^{n} \exp(\mathrm{sim}(q_i, p_{i,j}^-))}$$

Formal, Thibault et al. "SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval." *ArXiv* abs/2109.10086 (2021)

SPLADE Training

- Uses the same style contrastive loss as DPR

$$\mathrm{sim}(q,p) = w_q^\top w_p$$

$$L_{rank}(q_i, p_i^+, p_{i,1}^-, p_{i,2}^-, ..., p_{i,n}^-) = -\log \frac{\exp(\mathrm{sim}(q_i, p_i^+))}{\exp(\mathrm{sim}(q_i, p_i^+)) + \sum_{j=1}^{n} \exp(\mathrm{sim}(q_i, p_{i,j}^-))}$$

- Combined with **sparsity loss**es for query and passage representations

$$L = L_{rank}(q_i, p_i^+, p_{i,1}^-, p_{i,2}^-, ..., p_{i,n}^-) + \lambda_q L_{sparse}(w_q) + \lambda_d L_{sparse}(w_d)$$

Formal, Thibault et al. “SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval.” *ArXiv* abs/2109.10086 (2021)

SPLADE Training

- Uses the same style contrastive loss as DPR

$$\mathrm{sim}(q, p) = w_q^\top w_p$$

$$L_{rank}(q_i, p_i^+, p_{i,1}^-, p_{i,2}^-, \ldots, p_{i,n}^-) = -\log \frac{\exp(\mathrm{sim}(q_i, p_i^+))}{\exp(\mathrm{sim}(q_i, p_i^+)) + \sum_{j=1}^{n} \exp(\mathrm{sim}(q_i, p_{i,j}^-))}$$

- Combined with **sparsity losses** for query and passage representations

Tuneable hyperparameters that control sparsity strength

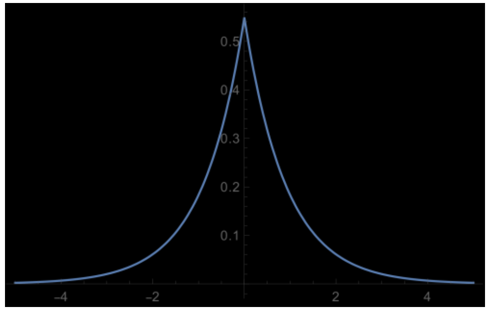
$$L = L_{rank}(q_i, p_i^+, p_{i,1}^-, p_{i,2}^-, \ldots, p_{i,n}^-) + \lambda_q L_{sparse}(w_q) + \lambda_d L_{sparse}(w_d)$$

Formal, Thibault et al. “SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval.” *ArXiv* abs/2109.10086 (2021)

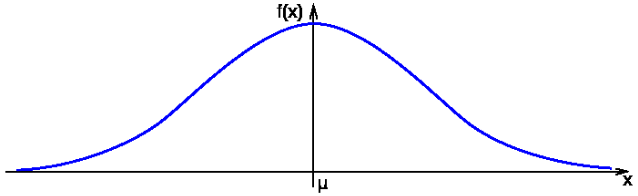
Sparsity Loss (vanilla)

- Common sparsity losses are L1/L2 regularization loss

$$L_1(w) = \sum_i |w_i|$$



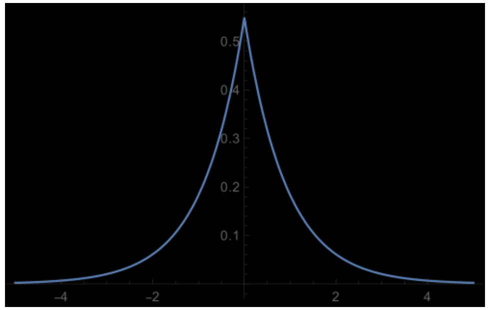
$$L_2(w) = \sum_i w_i^2$$



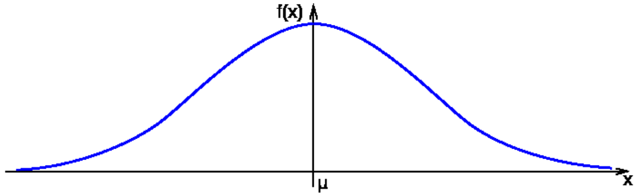
Sparsity Loss (vanilla)

- Common sparsity losses are L1/L2 regularization loss

$$L_1(w) = \sum_i |w_i|$$



$$L_2(w) = \sum_i w_i^2$$

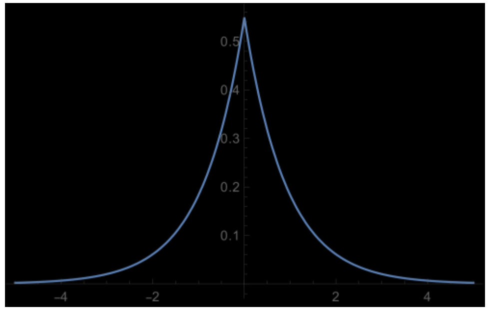


$$\lambda L_2(w) \propto -\log \prod_i \mathcal{N}(w_i : 0, \sigma^2) = -\sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}} \frac{1}{\sigma^2} \exp\left(-\frac{w_i^2}{2\sigma^2}\right)\right) =$$
$$n \log(\sqrt{2\pi}) + n \log \sigma^2 + \sum_i \frac{1}{2\sigma^2} w_i^2$$

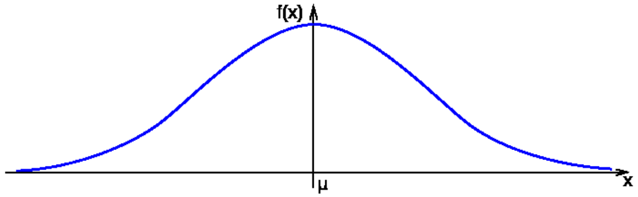
Sparsity Loss (vanilla)

- Common sparsity losses are L1/L2 regularization loss

$$L_1(w) = \sum_i |w_i|$$



$$L_2(w) = \sum_i w_i^2$$



$$\lambda L_2(w) \propto -\log \prod_i \mathcal{N}(w_i : 0, \sigma^2) = -\sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi}} \frac{1}{\sigma^2} \exp\left(-\frac{w_i^2}{2\sigma^2}\right)\right) =$$

$$\underbrace{n \log(\sqrt{2\pi}) + n \log \sigma^2} + \sum_i \frac{1}{2\sigma^2} w_i^2$$

Offsetting loss won't affect the optimization (`argmax [f(x)] = argmax [f(x)+c]`)

Sparsity Loss (complexity analysis)

- However, different representations might have same representations not sparse
  - L* losses only care about sparsity across vector dimension

p1	0	2	0	0	0	3
p2	0	4	0	0	0	8
p3	0	1	0	0	0	2

  - Expected # of FLOPS for sim(w_q,w_p) is **d/p**,
    where **d** is |w| **dimensionality** (V) and
    1/p is average **proportion** of non-0 elements in w

Biswajit Paria, Chih-Kuan Yeh, Ian E. H. Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos. 2020. Minimizing FLOPs to Learn Efficient Sparse Representations. arXiv:2004.05665

Sparsity Loss (complexity analysis)

- However, different representations might have same representations not sparse
  - L* losses only care about sparsity across vector dimension

p1	0	2	0	0	0	3
p2	0	4	0	0	0	8
p3	0	1	0	0	0	2

How to enforce sparsity across vectors?

Biswajit Paria, Chih-Kuan Yeh, Ian E. H. Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos. 2020. Minimizing FLOPs to Learn Efficient Sparse Representations. arXiv:2004.05665

FLOPS Sparsity Loss

- Estimate expected w_j across minibatch of M elements (p_1, p_2, …, p_M)
- Enforce sparsity of such estimate
- Expected # of FLOPS for sim(w_q,w_p) is **d/p^2**,
  where **d** is |w| **dimensionality** (V) and
  1/p is average **proportion** of non-0 elements in w

$$\bar{a}_j = \frac{1}{M}\sum_{i=0}^{M} w_j^{(p_i)} \qquad L_{FLOPS} = \sum_{j \in V} \bar{a}_j^2$$

- The probability that n-th element is sparse is the same for all n

Biswajit Paria, Chih-Kuan Yeh, Ian E. H. Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos. 2020. Minimizing FLOPs to Learn Efficient Sparse Representations. arXiv:2004.05665

SPLADE Training

- From the paper

3.4 Distillation and hard negatives

We also incorporate distillation to our training procedure, following the improvements shown in [12]. The distillation training is done in two steps: (1) we first train both a SPLADE first-stage retriever as well as a cross-encoder reranker [1] using the triplets generated by [12]; (2) in the second step, we generate triplets using SPLADE trained with distillation (thus providing harder negatives than BM25), and use the aforementioned reranker to generate the scores needed for the Margin-MSE loss. We then train a SPLADE model from scratch using these triplets and scores. The result of the second step is what we call DistilSPLADE-max.

Formal, Thibault et al. "SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval." *ArXiv* abs/2109.10086 (2021)

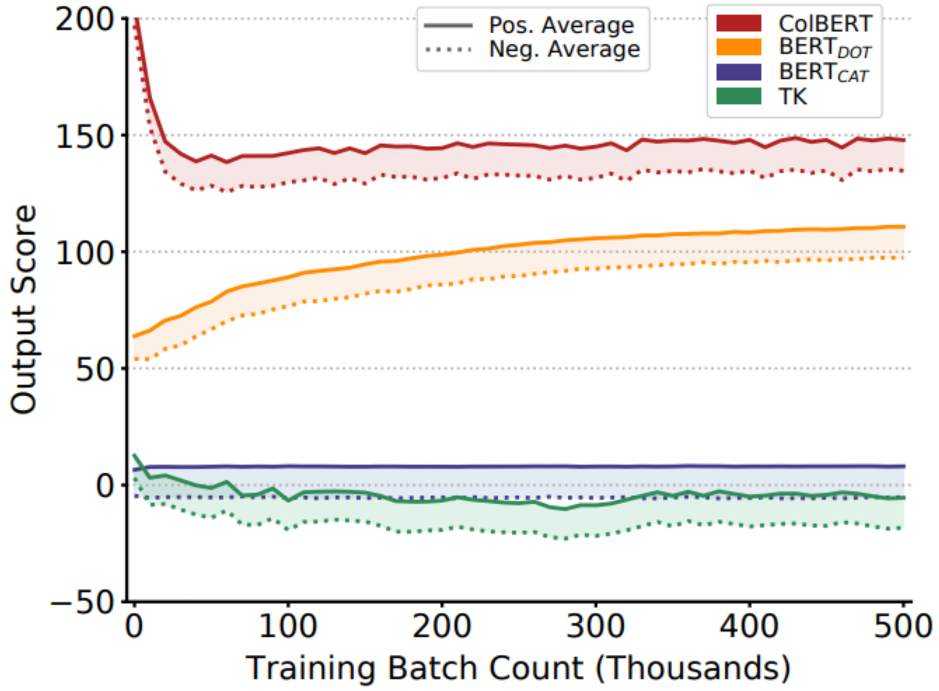
DistilSPLADE-max Training

1. Get triples for MS-MARCO from traditional system, picking negatives at random from top-1000 ranked passages. Construct first dataset of triplets (D#1)
2. Train Cross-Encoder on D#1 (CE#1).
3. Train SPLADE#1 on D#1 using **distillation** from CE#1.
4. Generate new negatives from SPLADE #1 top-K retrieved; construct new dataset of triplets (D#2).
5. Train Cross-Encoder on D#2 (CE#2).
6. Train SPLADE#2 on D#2 using **distillation** from CE#2.

Formal, Thibault et al. "SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval." *ArXiv* abs/2109.10086 (2021)

Distillation via Margin-MSE
Margin Mean Squared Error Loss

- Assume **Teacher Model** score function M_T.
  - E.g. BERT output projected to a scalar, as in Cross-Encoder.
- And **Student Model** score function M_S.
  - E.g., DPR/Contriever/SPLADE model's dot-product.
- To avoid enforcing specific scores from M_T to M_S **only margins** can be distilled from Teacher into Student.



Notes:

- Model Distillation = Transferring information between two models, training Student Model from Teacher Model.
- Score function = Similarity function

Hofstätter, Sebastian, et al. "Improving efficient neural ranking models with cross-architecture knowledge distillation." *arXiv preprint arXiv:2010.02666* (2020).

Distillation via Margin-MSE

Margin Mean Squared Error Loss

- Assume **Teacher Model** score function M_T.
  - E.g. BERT output projected to a scalar, as in Cross-Encoder.
- And **Student Model** score function M_S.
  - E.g., DPR/Contriever/SPLADE model's dot-product.
- Loss function for triples of queries Q, positive passages P^+, negative passages P^-.

$$\mathrm{L}(Q, P^+, P^-) = \mathrm{MSE}(\underbrace{\mathrm{M_s}(Q, P^+) - \mathrm{M_s}(Q, P^-)}_{\text{Student's error margin}}, \overbrace{\mathrm{M_t}(Q, P^+) - \mathrm{M_t}(Q, P^-)}^{\text{Teacher's error margin}})$$

$$\mathrm{MSE}(s, t) = \frac{1}{|t|} \sum_{i=0}^{|t|-1} (s_i - t_i)^2$$

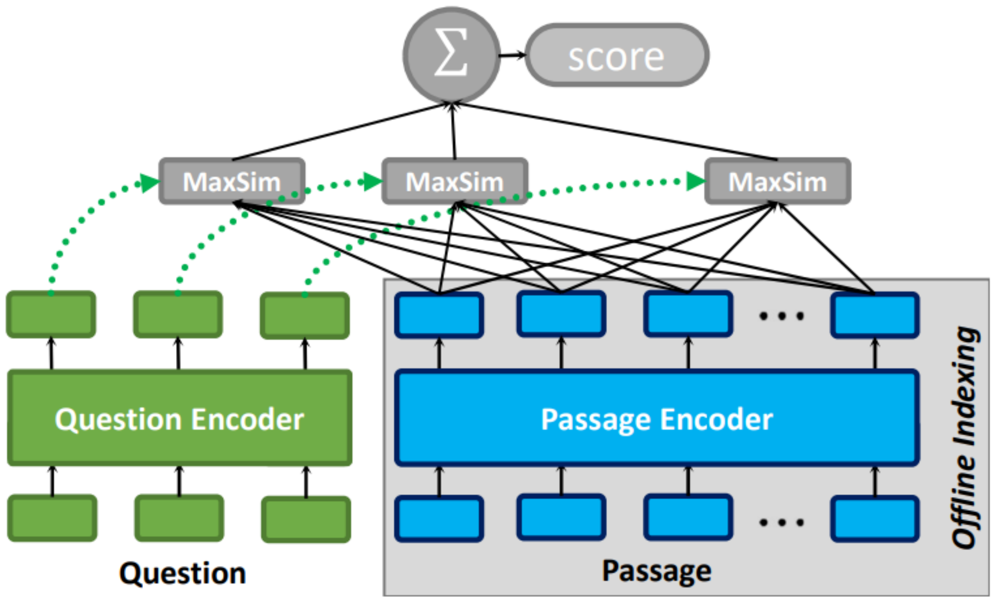
Hofstätter, Sebastian, et al. "Improving efficient neural ranking models with cross-architecture knowledge distillation." *arXiv preprint arXiv:2010.02666* (2020).

SPLADE Inference

- Similar to DPR/Contriever.
- Sparse-vector products, are efficiently implemented in Numba/Numpy.

Formal, Thibault et al. "SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval." *ArXiv* abs/2109.10086 (2021)

COLBERT V2



- **Multi-vector** query/document representations.
- Middle ground **between** cross-encoder and bi-encoder.
- Can be used for (first-stage) **retrieval**.
- However, it’s large passage index makes it suited for **smaller collections**.

Santhanam, Keshav, et al. "PLAID: an efficient engine for late interaction retrieval." *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022.

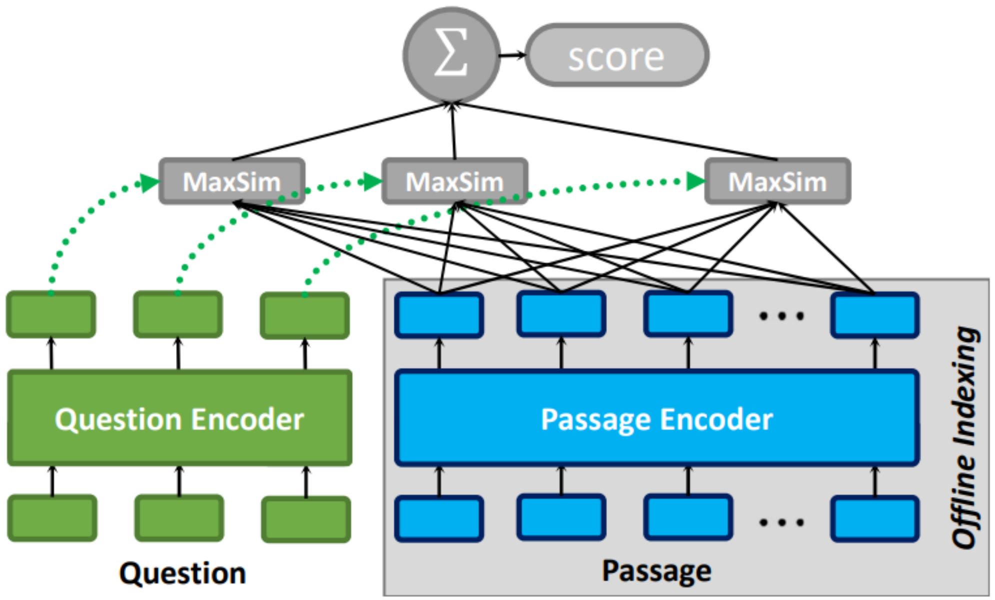
COLBERT V2

- Punctuation symbol embeddings are removed.
- Query is always represented with N_q(=32) tokens. Shorter queries are padded.
- Documents are split so that they contain N_d(=300 on BEIR) representations.
- Representations are low-domensional
  - (Each DPR vector has 768d, each of COLBERT's vectors is 128d).
- Each vector representation is L2 normalized (= unit

Santhanam, Keshav, et al. "PLAID: an efficient engine for late interaction retrieval." *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022.

COLBERT

MaxSim

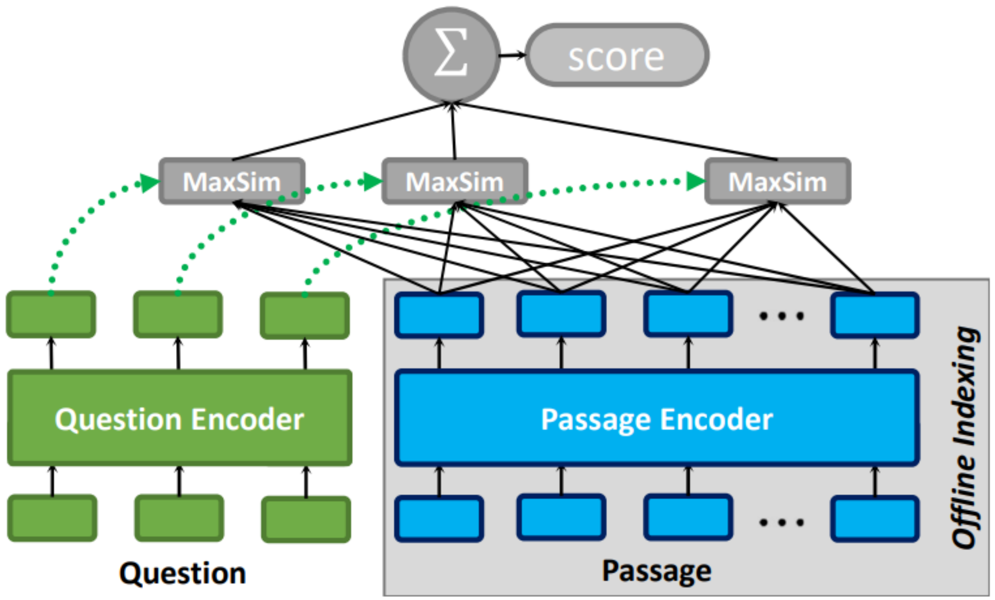


- Each token in Question/Passage is encoded into vector using BERT.
- Similarity between query and passage is computed between two matrices **Q** and **P**.
- Max-pooled **over document** representations.

Santhanam, Keshav, et al. "PLAID: an efficient engine for late interaction retrieval." *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022.

COLBERT

MaxSim



- Each token in Question/Passage is encoded into vector using BERT.
- Similarity between query and passage is computed between two matrices **Q** and **P**.
- Max-pooled **over document** representations.

$$sim(Q,P) = \sum_{i=0}^{N_q} \max_{j=0}^{N_d} \underbrace{Q_i^\top P_j}_{\text{Cosine Similarity}}$$

COLBERT Training

- Similar to previous methods.
- V2 uses hard negatives, cross-encoder distillation, in-batch negatives.

Santhanam, Keshav, et al. "PLAID: an efficient engine for late interaction retrieval." *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022.

COLBERT Inference (PLAID)

- Tomorrow practical

MTEB Massive Text Embedding Benchmark

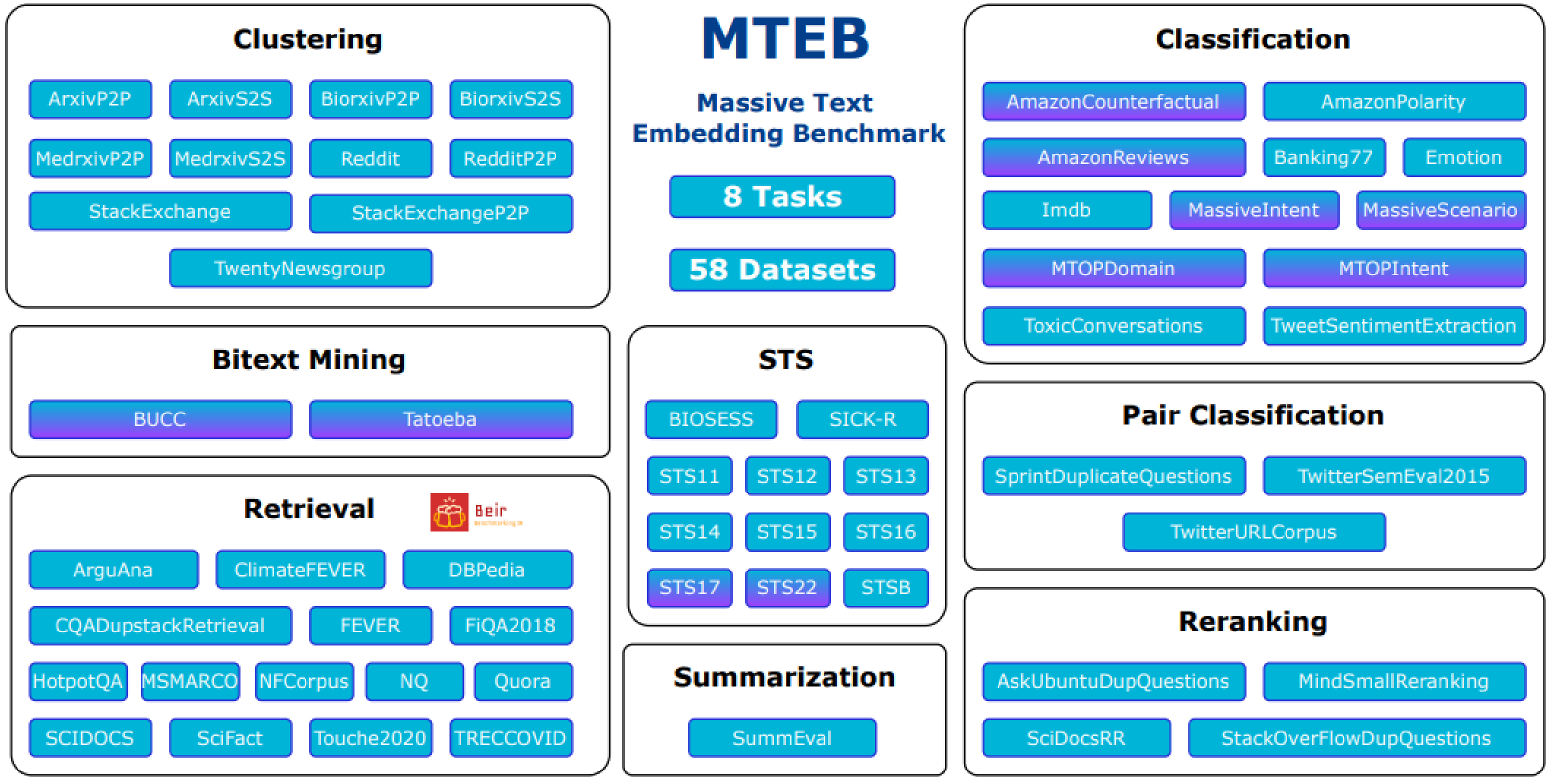


Figure 1: An overview of tasks and datasets in MTEB. Multilingual datasets are marked with a purple shade.

- https://huggingface.co/spaces/mteb/leaderboard

Muennighoff, Niklas, et al. "MTEB: Massive Text Embedding Benchmark." *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*. 2023.