Representation Learning

Nils Reimers
Principal Scientist @ cohere.ai
www.nils-reimers.de
My Career Path

- Ph.D. + PostDoc
- Neural Search Science Team
  - Team Lead
- Principal Scientist / Director of Machine Learning
  - Using very Large Language Models for search
Sentence Embeddings Model

How to learn Python?

[0.1, 0.2, …], [0.8, 0.5], …

BERT

Pooling

[0.4, 0.3, 0.7 … ]

Contextualized Word Embeddings

Fixed Sentence Embeddings

Neural Search – Bi-Encoders

Score

Cosine Similarity

vector

vector

pooling

pooling

BERT

BERT

Query

Document

Relevant Document

Query
Neural Search – Bi-Encoders

- Can overcome the lexical gap
  - US vs USA vs United States

- Respects the word order
  - Visa from Germany to Canada
  - Visa from Canada to Germany

- Knows about related terms
  - “spearman correlation numpy” finds the entry: “spearman correlation SciPy”
Multi-Modal & Multi-Lingual Search

- A cat on a table
- Two dogs in the snow
- London at night
- Image Embedding
- Text Embedding

Dos perros en la nieve
两只狗在雪中
Zwei Hunde im Schnee
Two dogs in the snow
Multiple Negative Ranking Loss

- Have positive pairs:
  \((a_1, p_1)\)
  \((a_2, p_2)\)
  \((a_3, p_3)\)

- Examples:
  - (query, answer-passage)
  - (question, duplicate_question)
  - (paper title, cited paper title)

- \((a_i, p_i)\) should be close in vector space

- \((a_i, p_j)\) should be distant in vector space \((i \neq j)\)
  - Unlikely that e.g. two randomly selected questions are similar

- Also called “training with in-batch negatives”, InfoNCE or NTXentLoss
Multiple Negative Ranking Loss

- Mathematical Definition

$$L = -\frac{1}{n} \sum_{i=1}^{n} \frac{\exp(sim(a_i, p_i))}{\sum_j \exp(sim(a_i, p_j))}$$

- Sim: Similarity function between (a, p)
  - Cosine-Similarity
  - Dot-Product
Intuitive Explanation

- $a_1$: How many people live in Berlin?
  - $p_1$: Around 3.5 million people live in Berlin
  - $p_2$: Washington DC is the capital of the US
  - $p_3$: The 2021 Olympics are held in Japan

- Compute text embeddings & compute similarities:
  - $\text{sim}(a_1, p_1) = 0.5$
  - $\text{sim}(a_1, p_2) = 0.3$
  - $\text{sim}(a_1, p_3) = 0.1$

- See it as classification task and use Cross-Entropy Loss:
  - Prediction: [0.5, 0.3, 0.1]
  - Gold: [1, 0, 0]
Multiple Negative Ranking Loss
Intuitive Explanation

- \(a_1: \) How many people live in Berlin?, \(p_1: \) Around 3.5 million people live in Berlin
- \(a_2: \) What is the capital of the US?, \(p_2: \) Washington DC is the capital of the US
- \(a_3: \) Where are the Olympics this year?, \(p_3: \) The 2021 Olympics are held in Japan

- Compute text embeddings & compute similarities:
  \[
  \text{sim}(\text{vec}_a, \text{vec}_b) = \text{vec}_a \ast \text{vec}_b^T = \begin{bmatrix}
  \text{sim}(a_1, p_1), \text{sim}(a_1, p_2), \text{sim}(a_1, p_3) \\
  \text{sim}(a_2, p_1), \text{sim}(a_2, p_2), \text{sim}(a_2, p_3), \\
  \text{sim}(a_3, p_1), \text{sim}(a_3, p_2), \text{sim}(a_3, p_3)
  \end{bmatrix}
  \]

- See it as classification task and use Cross-Entropy Loss:
  - Gold:
    \[
    \begin{bmatrix}
    1, & 0, & 0, \\
    0, & 1, & 0, \\
    0, & 0, & 1
    \end{bmatrix}
    \]
Multiple Negatives Ranking Loss

Code

```python
scores = self.similarity_fct(embeddings_a, embeddings_b) * self.scale
labels = torch.tensor(range(len(scores)), dtype=torch.long, device=scores.device)  # Example a[i] should match with b[i]
return self.cross_entropy_loss(scores, labels)
```

https://github.com/UKPLab/sentence-transformers/blob/master/sentence_transformers/losses/MultipleNegativesRankingLoss.py
Multiple Negatives Ranking Loss

Similarity Functions

- **How to compute \(\text{sim}(a, b)\)?**
  - \(a, b\) are vectors

  - **Dot-product:** \(\text{dot\_prod}(a, b) = ab^T\)

  - **Cosine-Similarity:** \(\text{cos\_sim}(a, b) = (ab^T) / (||a|| \cdot ||b||)\)
    - Does not work well, scores differences are too small

  - **Scaled Cosine-Similarity:** \(\text{scaled\_cos\_sim}(a, b) = C \cdot \text{cos\_sim}(a, b)\)
    - Works well with e.g. \(C=20\)

  - **Scaled dot-product:** \(\text{scaled\_dot\_prod}(a, b) = C \cdot \text{dot\_prod}(a, b)\)
## Cosine-Similarity vs. Dot-Product

<table>
<thead>
<tr>
<th>Cosine-Similarity</th>
<th>Dot-Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Vector has highest similarity to itself</td>
<td>▪ Other vectors can have higher dot-product</td>
</tr>
<tr>
<td>▪ cos_sim(a, a) = 1</td>
<td>▪ dot(a, a) &lt; dot(a, b)</td>
</tr>
<tr>
<td>▪ With normalized vectors, equal to dot_product</td>
<td>▪ Might be slower with certain approximate nearest neighbor methods</td>
</tr>
<tr>
<td>▪ With max vector length = 1</td>
<td>▪ Max vector length not know</td>
</tr>
<tr>
<td>▪ With normalized vectors, proportional to Euclidian distance</td>
<td>▪ Does not work with k-means clust.</td>
</tr>
<tr>
<td>▪ Works with k-means clustering</td>
<td></td>
</tr>
</tbody>
</table>

https://arxiv.org/abs/2104.08663  
Cosine-Similarity vs. Dot-Product

- Semantic search: Given short query, find longer passage
- Cosine-Similarity: Prefers retrieval of short passages close to query
- Dot-Product: Prefers longer passages (longer passage = longer vector = higher dot product)

https://arxiv.org/abs/2104.08663
Optimizing the Multiple-Negatives-Ranking-Loss

- Training with $\text{scaled\_cos\_sim}(a, b) = C \times \text{cos\_sim}(a, b)$
  - How to choose the scale $C$? <= unclear, common values 14-20
  - ConveRT paper: Start at 1, end at 23, increase over first 10k steps
  - CLIP paper: $\text{scaled\_cos\_sim}(a, b) = \exp(C) \times \text{cos\_sim}(a, b)$ with $C$ a learnable parameter
  - Unclear impact
    - Will it make a difference?
    - Does it depend on the data / task?

- Symmetric Multiple-Negatives-Ranking-Loss
  - Used in CLIP Paper
  - Compute: $(\text{Loss}(A, P) + \text{Loss}(P, A)) / 2$
  - Swap anchor & positives (e.g. given answer, what is the question?)
  - Unclear impact
Multiple-Negatives-Ranking-Loss with Additive Margin

\[ \text{sim}(a_i, p_j) = \begin{cases} 
\text{sim}(a_i, p_i) - m & \text{if } i = j \\
\text{sim}(a_i, p_j) & \text{otherwise}
\end{cases} \]

- Substract value \( m \) from positive pairs
  - Consine-similarity with margin 0.3 used in LaBSE paper with translation pairs
- Unclear impact of margin for other tasks / datasets

Multiple Negative Ranking Loss
Hard Negatives

- Larger batch size => task more difficult => better results
  - Given query, which of the 10 passages provide the answer?
  - Given query, which of the 1k passages provide the answer?

Multiple Negative Ranking Loss

Hard Negatives

- Train with tuples:
  \((a_1, p_1, n_1)\)
  \((a_2, p_2, n_2)\)

- \(n_i\) should be similar to \(p_i\) but not match with \(a_i\)

- Bad example:
  - a:  How many people live in London?
  - p:  Around 9 million people live in London
  - n:  London has a population of 9 million people.

- Good example:
  - a:  How many people live in London?
  - p:  Around 9 million people live in London
  - n:  Around 1 million people live in Birmingham, second to London.
How to find hard-negatives?

- Quality of hard-negatives significantly improves the performance
- Finding good hard negatives not easy

**Strategy 1: Exploit structure in your data**
- Citation graph: (Title, Cited_Paper, Paper_Cited_by_Cited_Paper)
- Q&A: (Question, Answer with many stars, Answer with few stars)

**Strategy 2: Mine hard negative:**
- Use BM25 to find top-100 most similar texts to anchor / positive
- Select one of these randomly
- Make sure that these are actually negatives!
Improving Quality with Better Batches

- Assume you have (question, answer) pairs from StackExchange
  - 140 different subforums: StackOverflow, Travel, Cooking, ...

- Naïve approach:
  - Randomly sample data from all pairs:
    - [ (question_python, answer_python),
      (question_travel, answer_travel),
      (question_pasta, answer_pasta) ]

- Finding the right answer for a given question is easy
  - Question about Python => Take that one programming answer in the batch...
Assume you have (question, answer) pairs from StackExchange
- 140 different subforums: StackOverflow, Travel, Cooking, ...

Better approach
- Sample pairs from one subforum (e.g. StackOverflow)
  [ (question_python, answer_python),
    (question_java, answer_java),
    (question_c, answer_c)]
Improving Quality with Better Batches

- Assume you have (question, answer) pairs from StackExchange
  - 140 different subforums: StackOverflow, Travel, Cooking, ...

- Even better approach (?)
  - Sample pairs from same / similar tags (e.g. StackOverflow, Python tag)
    - (question_python, answer_python),
    - (question_numpy, answer_numpy),
    - (question_pandas, answer_pandas)

- Adding random batches might still be needed
  - Otherwise StackOverflow vector space could overlap with Travel vector space
  - 90% difficult batches, 10% easy random batches
  - Or: start with mainly random batches, then go to difficult batches
Bi-Encoders and the Curse of the Unknowns

• How do Bi-Encoders handle unknown words?
  • Not seen during pre-training
  • Not seen during fine-tuning

• Where to put these words in a vector space?
  • XLNet
  • Clexchain
  • Forwrd
  • 0xc004f213

• How to know
  • Corona Virus ⇔ COVID-19 ⇔ SARS-Cov-2
  • Q: “Which vision transformer model is the best?”
    A: “ViT has been doing great in our experiments”
Challenge of Unknown Words for Dense Bi-Encoders

“BigBirdPegasus”

Dense Encoder

1) bigboss
2) bigdata
3) bigass

![Image](https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/)
Unknown Words for Sparse Bi-Encoders

“BigBirdPegasus”

Split query in word pieces:
- big:2.1, ##bird: 2.0, ##pe: 1.8, ##gas: 2.0, ##us:1.9

Some related terms are added:
- ##birds: 1.2, giant: 0.7
Bi-Encoders vs Lexical Search

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BM25</th>
<th>Dense Model (TAS-B)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-Domain</td>
<td>22.8</td>
<td>40.8</td>
<td>+18.0</td>
</tr>
<tr>
<td>BioASQ</td>
<td>46.5</td>
<td>38.3</td>
<td>-8.2</td>
</tr>
<tr>
<td>SCIDOCS</td>
<td>15.9</td>
<td>14.9</td>
<td>-1.0</td>
</tr>
</tbody>
</table>

- BM25 was better on 10 / 18 datasets
Do Models Generalize?

- BM25 lexical search a strong baseline
- BM25 + CrossEncoder re-ranking perform the best
- Dense embedding models (TAS-B, ANCE, DPR) with issues for unknown domains
- Sparse embedding models (SPLADEv2) better for unknown domains
Cross-Encoders vs Bi-Encoders

![Graph comparing Cross-Encoders and Bi-Encoders in STSb (English)](https://arxiv.org/pdf/2010.08240.pdf)

The graph shows the Spearman rank correlation ($\rho$) for both Cross-encoder and Bi-encoder models as a function of training size in thousands ($k$). As the training size increases, the correlation also increases, indicating improved performance for both models. However, Cross-encoder models generally outperform Bi-encoder models across different training sizes.
## Cross-Encoders vs Bi-Encoders

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BM25</th>
<th>Dense Model (TAS-B)</th>
<th>BM25 + CE</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-Domain</td>
<td>22.8</td>
<td>40.8</td>
<td>41.3</td>
</tr>
<tr>
<td>BioASQ</td>
<td>46.5</td>
<td>38.3</td>
<td>52.3</td>
</tr>
<tr>
<td>CQADupStack</td>
<td>29.9</td>
<td>31.4</td>
<td>37.0</td>
</tr>
<tr>
<td>TREC-COVID</td>
<td>65.5</td>
<td>48.1</td>
<td>75.7</td>
</tr>
<tr>
<td>SCIDOCS</td>
<td>15.9</td>
<td>14.9</td>
<td>16.6</td>
</tr>
</tbody>
</table>

- BM25 + CE on average 13.8 points better than dense
Why not using Cross-Encoders / doc2query?

• Cross-Encoders are slow (even small ones)
  • E.g. query has 10 tokens, docs have 240 tokens, re-rank 100 docs
  • Bi-Encoders: Compute embedding for query (e.g. 10ms)
  • Cross-Encoder: Re-rank 100 x 250 token docs
    • Forward pass for 250 tokens takes ~25*25 = 625 times longer
    • Overall 62,500 times longer to get results

• Doc2query is slow at indexing
  • Generates 40 query per passage
  • Question generation is extremely slow
  • Costs to generate queries for 8M docs: $750
  • Computing dense embeddings: $1

How to Adapt Bi-Encoders to New Domains?

Adaptive Pre-Training

Pre-Training on Target Domain → Fine-Tuning on Labeled Data (MS MARCO)

❌ Requires (expensive) training on labeled source data

What we want:

Fine-Tuning on Labeled Data (MS MARCO) → Unsupervised adaptation on Target Domain

✔ Use pre-trained models and adapt to your domain
Adaptive Pre-Training

Pre-Training on Target Domain  
Fine-Tuning on Labeled Data (MS MARCO)

<table>
<thead>
<tr>
<th>Methods for Pre-Training</th>
<th>Does it work?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masked Language Modeling (MLM)</td>
<td>Yes</td>
</tr>
<tr>
<td>TSDAE</td>
<td>Yes</td>
</tr>
<tr>
<td>Inverse Cloze Task (ICT)</td>
<td>Yes</td>
</tr>
<tr>
<td>SimCSE</td>
<td>No – weaker than base model</td>
</tr>
<tr>
<td>Contrastive Tension (CT)</td>
<td>No – weaker than base model</td>
</tr>
<tr>
<td>Condenser (CD)</td>
<td>No – weaker than base model</td>
</tr>
</tbody>
</table>
Masked Language Model (MLM)
Python is a programming language. Its design philosophy emphasizes code readability. Python uses significant indentation.

Its design philosophy emphasizes code readability.

Select sentence at random

Python is a programming language. Python uses significant indentation.

Train such sentence + remaining paragraph are close in vector space

https://aclanthology.org/P19-1612.pdf
TSDAE

- Delete randomly words in the text
- Pass through the encoder
- Apply pooling to get fixed-sized text embedding
- Decoder must reconstruct text without noise from this text embedding

https://arxiv.org/abs/2104.06979
## Adaptive Pre-Training - Results

<table>
<thead>
<tr>
<th>Models</th>
<th>4 Sentence Tasks</th>
<th>6 Dense IR Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out-of-the-box</td>
<td>52.3</td>
<td>45.2</td>
</tr>
<tr>
<td><strong>Source -&gt; Target</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSDAE</td>
<td>54.2</td>
<td>-</td>
</tr>
<tr>
<td>MLM</td>
<td>51.1</td>
<td>-</td>
</tr>
<tr>
<td><strong>Target -&gt; Source</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSDAE</td>
<td>56.5</td>
<td>49.2</td>
</tr>
<tr>
<td>MLM</td>
<td>55.9</td>
<td>46.7</td>
</tr>
<tr>
<td>ICT</td>
<td>-</td>
<td>46.5</td>
</tr>
<tr>
<td>SimCSE</td>
<td>52.4</td>
<td>45.0</td>
</tr>
<tr>
<td>CD</td>
<td>-</td>
<td>44.7</td>
</tr>
<tr>
<td>CT</td>
<td>53.0</td>
<td>44.0</td>
</tr>
</tbody>
</table>

Domain Adaptation on Pre-Trained Model

❌ Adaptive pre-training is expensive
   1) Unsupervised training on target domain
   2) Fine-tuning on labeled source dataset (can be as large as 1B+ training pairs)

✔ What we want:
   1) Fine-tuning on labeled source dataset (can be as large as 1B+ training pairs)
   2) Unsupervised training on target domain

Generative Pseudo Labels (GPL) is able to achieve this

https://arxiv.org/abs/2112.07577
GPL – Generative Pseudo Labeling

Fine-tuned model (e.g. on MSMARCO) → GPL: Unsupervised domain adaptation

GPL:

Query Generation via T5 → Negative Mining via Dense Retrieval → Pseudo Labeling via Cross-Encoder

- Python is ...
- What is Python
- Java is ...
- What is Python
- Python is ...
- Java is ...
- What is Python
- Python is ...
- Java is ...
- What is Python

0.3 6.2 2.0
Step 1: Generate Queries

Python is a programming language. Its design philosophy emphasizes code readability. Python uses significant indentation.

What is Python?  How good is the readability of Python code?  Which programming language uses indentation?
Step 2: Mine Negatives

Which programming language uses indentation?

You can use indentation to structure your code

Java is a class-based programming language

C is one of the fastest programming language available
Step 3: Score Pairs with CrossEncoder

(QUERY, DOC1) -> 5.2
(QUERY, DOC2) -> 0.1
(QUERY, DOC3) -> -3.8
Why do we need the CrossEncoder?

- **Query asks for definition** of "futures contract"

- **Easy negatives**: Mention "futures contract" only

- **False negative**

- **Hard negative**: Give partial definition

<table>
<thead>
<tr>
<th>Item</th>
<th>Text</th>
<th>GPL</th>
<th>QGen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>what is futures contract</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Positive</td>
<td>Futures contracts are a member of a larger class of financial assets called derivatives ...</td>
<td>10.3</td>
<td>1</td>
</tr>
<tr>
<td>Negative 1</td>
<td>... Anyway in this one example the s&amp;p 500 futures contract has an &quot;initial margin&quot; of $19,250, meaning ...</td>
<td>2.0</td>
<td>0</td>
</tr>
<tr>
<td>Negative 2</td>
<td>... but the moment you exercise you must have $5,940 in a margin account to actually use the futures contract ...</td>
<td>0.3</td>
<td>0</td>
</tr>
<tr>
<td>Negative 3</td>
<td>... a futures contract is simply a contract that requires party A to buy a given amount of a commodity from party B at a specified price...</td>
<td>8.2</td>
<td>0</td>
</tr>
<tr>
<td>Negative 4</td>
<td>... A futures contract commits two parties to a buy/sell of the underlying securities, but ...</td>
<td>6.9</td>
<td>0</td>
</tr>
</tbody>
</table>
Train Bi-Encoder with MarginMSE-Loss

Compute Loss

$|s_p - s_n| \text{ vs } |ce_p - ce_n|$

Compute dot-scores

CrossEncoder teaches BiEncoder how far vectors are supposed to be in vector space

Compute Embeddings

Bi-Encoder

Query Positive Doc Negative Doc
Results

<table>
<thead>
<tr>
<th>Models</th>
<th>6 Dense IR Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out-of-the-box</td>
<td>45.2</td>
</tr>
<tr>
<td>Target -&gt; Source</td>
<td></td>
</tr>
<tr>
<td>TSDAE</td>
<td>49.2</td>
</tr>
<tr>
<td>MLM</td>
<td>46.7</td>
</tr>
<tr>
<td>Generative Pseudo Labeling</td>
<td></td>
</tr>
<tr>
<td>GPL</td>
<td>51.4</td>
</tr>
<tr>
<td>TSDAE+GPL</td>
<td>52.4</td>
</tr>
</tbody>
</table>
Multilingual Models
Translation Language Model

- Concatenate parallel data and run MLM

Previous Approaches: LASER

- Use output of encoder from translation system
- Issues:
  - Cannot control what type of embeddings are learned
  - Works poorly on identifying similar sentences
Previous Approaches: Multilingual USE

- Multi-task setup with bridging task
- Issues:
  - Getting bridging task right is challenging + requires large batch sizes
  - Hard to extend model afterwards to new languages
LaBSE

• Pre-Training
  • Trained on large mono-lingual dataset via MLM
  • Trained on translation pairs via TLM (Translation Lang. Model)
• Fine-tuned on translation pairs via MultipleNegativesRankingLoss

Multilingual Knowledge Distillation

• Given:
  • Teacher sentence embedding model T (e.g. SBERT trained on English STS)
  • Parallel sentence data \(((s_1, t_1), \ldots, (s_n, t_n))\)
  • Student model S with multilingual vocabulary (e.g. XLM-R + Mean Pooling)

• Train student S such that:

\[
S(s_i) \approx T(s_i) \\
S(t_i) \approx T(s_i)
\]

## Performance

<table>
<thead>
<tr>
<th>Model</th>
<th>STS</th>
<th>BUCC Bitext Mining</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge Distillation</td>
<td>83.7</td>
<td>88.6</td>
</tr>
<tr>
<td>mUSE</td>
<td>81.1</td>
<td>87.7</td>
</tr>
<tr>
<td>LaBSE</td>
<td>73.5</td>
<td>93.5</td>
</tr>
<tr>
<td>LASER</td>
<td>67.0</td>
<td>93.0</td>
</tr>
</tbody>
</table>

- Models strong on multilingual STS are weak on Bitext Mining
  - Knowledge Distillation / mUSE puts similar sentences close, but which are not perfect translations

Language Bias

- Preference of certain language combinations
- Language bias impacts performance negatively on multilingual pools
- LASER and LaBSE with strong language bias

<table>
<thead>
<tr>
<th>Model</th>
<th>Expected Score</th>
<th>Actual Score</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>LASER</td>
<td>69.5</td>
<td>68.6</td>
<td>-0.92</td>
</tr>
<tr>
<td>mUSE</td>
<td>81.7</td>
<td>81.6</td>
<td>-0.19</td>
</tr>
<tr>
<td>LaBSE</td>
<td>74.4</td>
<td>73.1</td>
<td>-1.29</td>
</tr>
<tr>
<td>XLM-R ← SBERT-paraphrases</td>
<td>84.0</td>
<td>83.9</td>
<td>-0.11</td>
</tr>
</tbody>
</table>

Language Bias – Good or Bad?

Side-effects **with** language bias:

• Same language results are ranked higher just because of language
• There might be better hits / answers in other languages
Side-Effects **without** Language Bias

**wedding**

Who is the president?
A: Joe Biden is the current president

**शादी** (hindi: wedding)

qui est le président?
A: Joe Biden is the current president
Multilingual Search Models

- Should same language results be preferred?
  - Yes: Language Bias (weak alignment)
  - No: Strong alignment

Batch Strategy

- X-X is a bad idea
  - Easy to find the correct answer (just check for language)

- X-X-mono best when Q & A in same language

- X-Y best when Q & A can be in different languages

---

Figure 3: Sample batches for each baseline.
Conclusion

- Training for bitext mining models:
  - LaBSE

- Training for **cross-lingual search** model
  - X-Y batch strategy
  - Getting large scale cross-lingual data is difficult

- Training for **multilingual search** models
  - X-X-mono batch strategy
  - Train on (multilingual) pair & triplet datasets
  - Add parallel data for alignment
Cross-Encoder
Cross-Encoder

- Concatenate: *Query [SEP] Passage*
- Map CLS-token output to single score
Cross-Encoder

1. Stage Retrieval

- Document Collection
- Retrieve candidates
- Cross-Encoder

Search Query / Question

Re-Ranker

Ranked hits

Cross-Encoder

0...1

Classifier

BERT

Query

Document

BERT Classifier
Learning to Rank – Pointwise Loss

- **Pointwise-Loss**
  - Given (query, document, label) triplets
  - Set label=0 / label=1 for non-/relevant docs
  - Binary classification task: $\text{BCELoss}(\text{CE(Query, Doc)}, \text{Label})$

- **Challenges:**
  - How many non-relevant to relevant docs in the training?
  - Relevance is not black & white
Learning to Rank – Pairwise Loss

- Given (query, doc1, doc2) triplets
- Is doc1 or doc2 more relevant to the query?
- For simplification: Assume doc1 is more relevant than doc2

**RankNet Loss:**
- Compute scores: $s^+ = CE(query, doc^+)$, $s^- = CE(query, doc^-)$
- Loss(query, doc+, doc-) = BCELoss($s^+ - s^-$, 1) = $\log(\text{sigmoid}(s^+ - s^-))$
- We try to maximize the margin between $s^+$ and $s^-$

- We don’t need absolute relevance labels, just relative preferences (A or B)
- Works nice with click logs / transaction logs: Given query, what was clicked
Learning to Rank – Listwise Loss

- Given (query, doc₁, doc₂, doc₃, ...)
- Which doc is the most relevant for the query?
- Many loss functions available: LambdaRank, LambdaMART, ApproxNDCG, NeuralNDCG...
  - Often they try to optimize the eval measure (like nDCG)
  - I didn’t observe large differences
- I prefer ListRank Loss / ListNet Loss:
  - Compute \( s₁ = CE(query, doc₁), s₂ = CE(query, doc₂), s₃ = CE(query, doc₃), ... \)
  - CrossEntropyLoss( \([s₁, s₂, s₃, ...], label\) )
  - Label: Which document is the most relevant?
  - Train with 1 positive and many negative docs
  - With 1 negative: Identical to Pairwise Loss / RankNet Loss
Learning to Rank

- Pointwise loss performs the worse
  - Hard to tell what docs are relevant / irrelevant
  - Hard to select the ratio of positive vs negative labels
  - Harder to get labeled data

- Pairwise / Listwise Loss performs better
  - Just relative importance is relevant (is A or B better?)
  - Easier to extract from click logs / transaction logs
Importance of Negatives

- Listwise loss: [query, positive, neg1, neg2, ...]
- Negatives are either from top-50 or top-1k from BM25
Importance of Negatives

<table>
<thead>
<tr>
<th>Train Neg↓ / Inference→</th>
<th>BM25</th>
<th>BM25*</th>
<th>HDCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25</td>
<td>39.6</td>
<td>39.5</td>
<td>38.1</td>
</tr>
<tr>
<td>BM25*</td>
<td>40.7</td>
<td>42.3</td>
<td>41.8</td>
</tr>
<tr>
<td>HDCT</td>
<td>40.8</td>
<td>41.9</td>
<td>43.4</td>
</tr>
</tbody>
</table>

- Performance drops if train sample is different from test first-stage retrieval system
- As we optimize for unknown first-stage system:
  - Samples negatives from different systems (lexical & embedding based)
Number of Negatives

![Graph showing number of negatives over time](https://arxiv.org/pdf/2101.08751.pdf)

- 1 negative
- 3 negatives
- 5 negatives

## Multilingual Cross-Encoder

- Trained on Machine-Translated MS MARCO (incl. de, ar, id, ru)
- Performance on GermanQuAD & Mr. Tydi (Arabic, Indonesian, Russian)

<table>
<thead>
<tr>
<th>Model</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>mdeberta v3 (training: English only)</td>
<td>52.2</td>
</tr>
<tr>
<td>mdeberta v3 (14 mMARCO langs)</td>
<td>53.0</td>
</tr>
<tr>
<td>LaBSE</td>
<td>52.6</td>
</tr>
<tr>
<td>mMiniLM</td>
<td>52.0</td>
</tr>
</tbody>
</table>

- Models perform surprisingly well even when train on English only