Representation Learning

Nils Reimers
Principal Scientist @ cohere.ai

www.nils-reimers.de

co:here

My Career Path

UBIQUITOUS KNOWLEDGE PROCESSING

Ph.D. + PostDoc

Neural Search Science Team
Team Lead

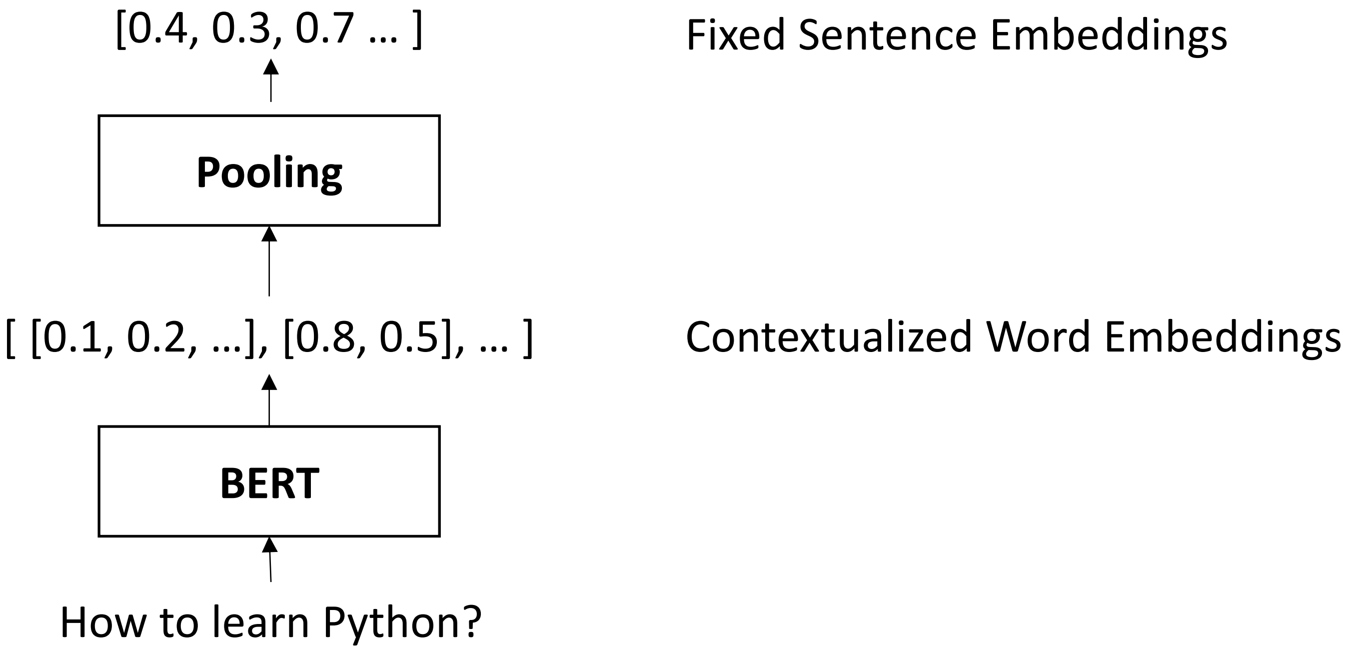
sentence-transformers

TSDAE

GPL

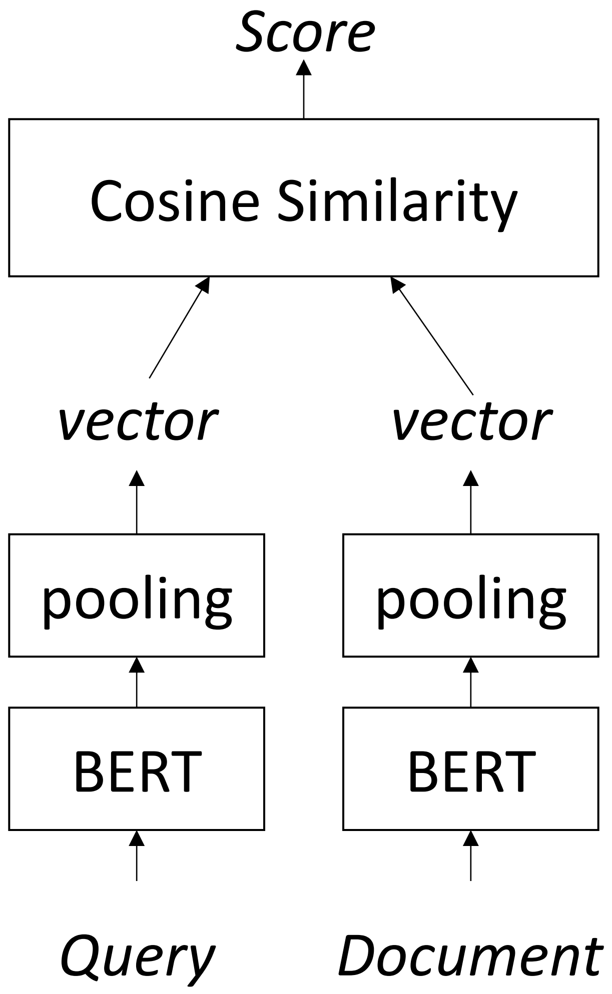
Principal Scientist / Director of Machine Learning
Using very Large Language Models for search

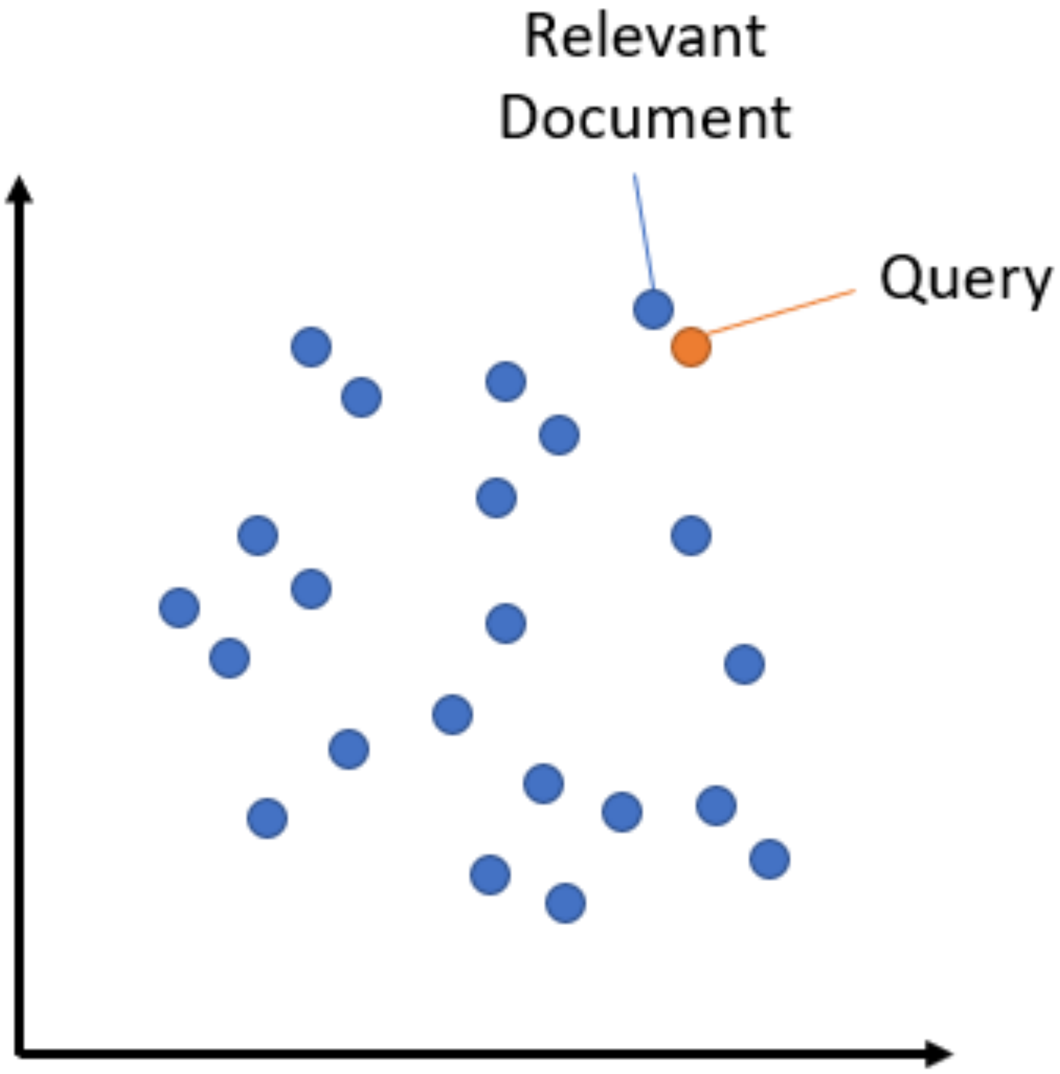
Sentence Embeddings Model



Nils Reimers, Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. EMNLP 2019

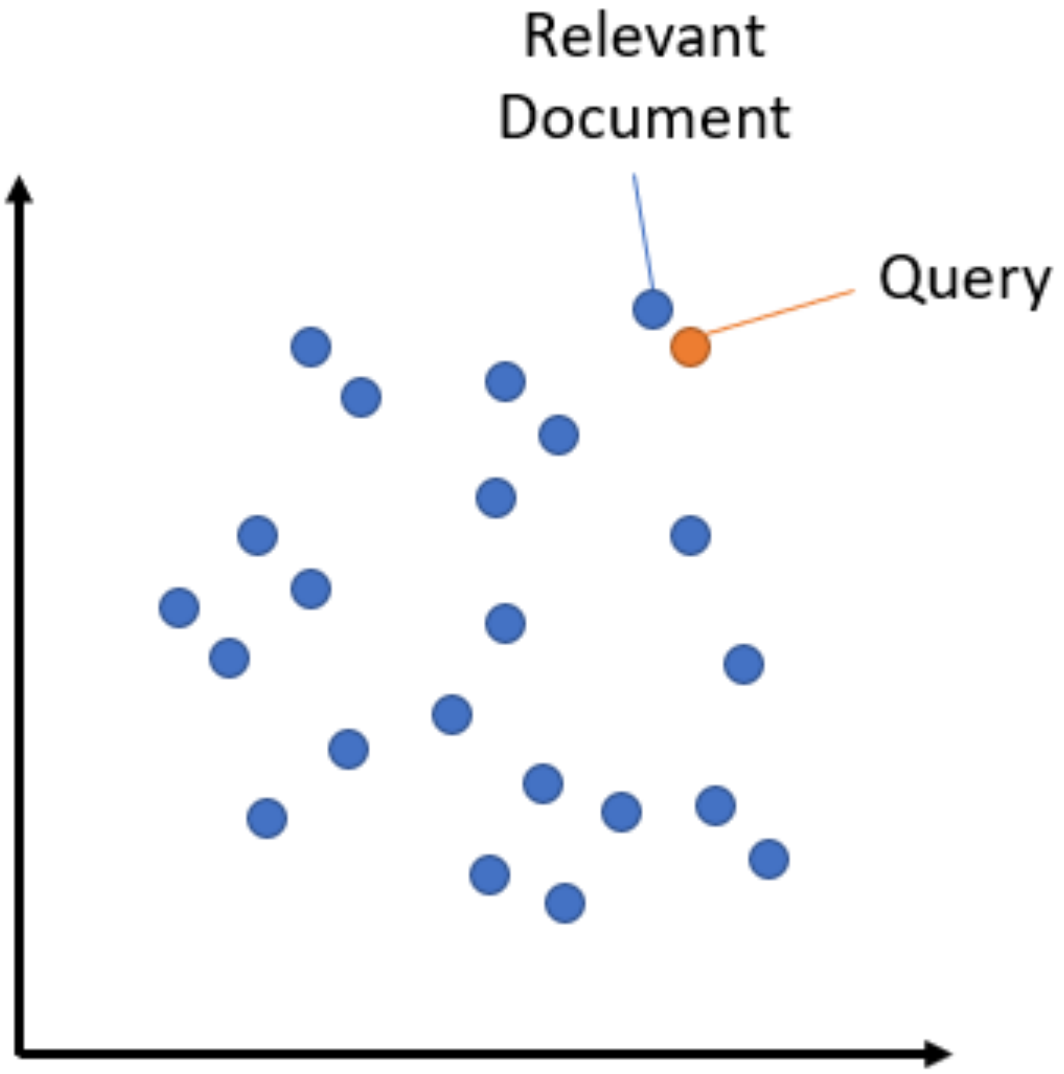
Neural Search – Bi-Encoders



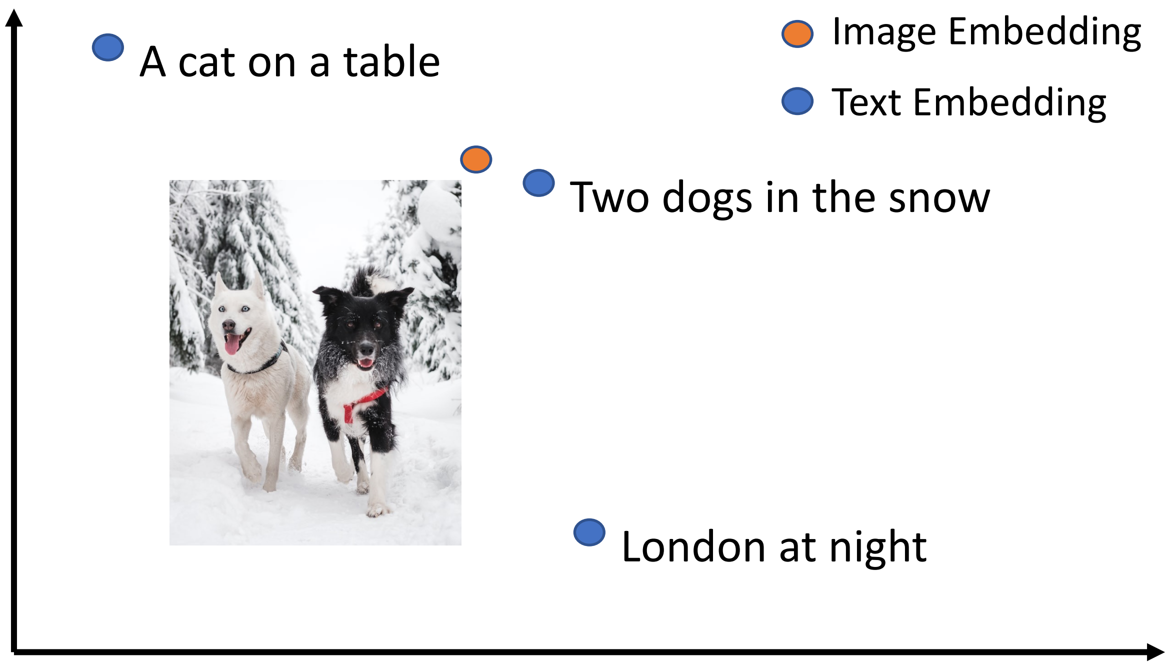


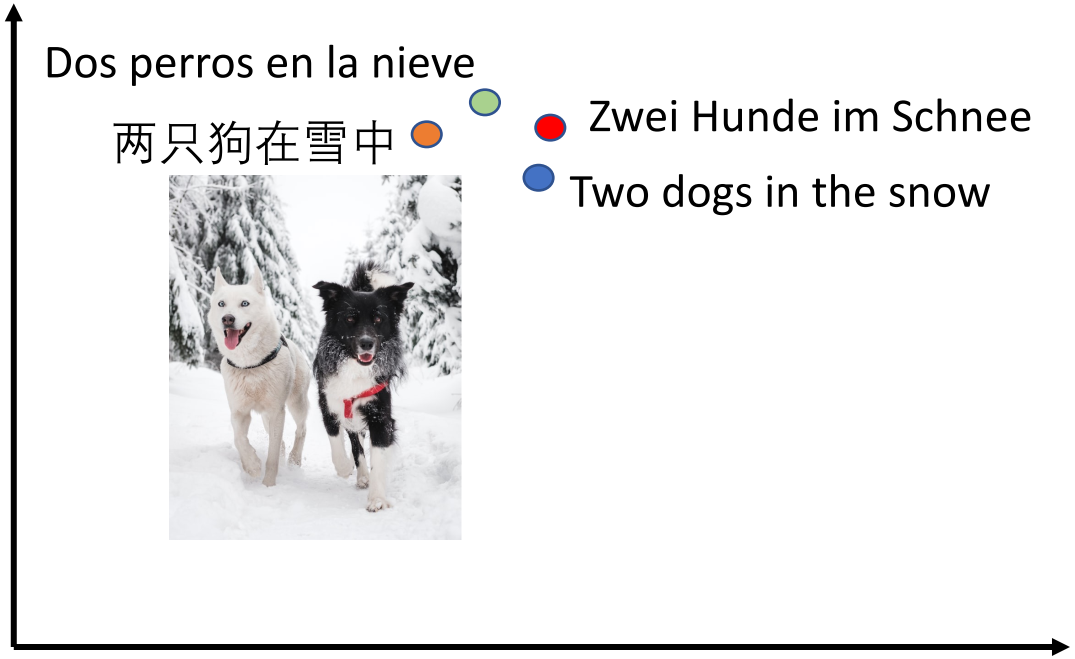
Neural Search – Bi-Encoders

- 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





Multiple Negative Ranking Loss

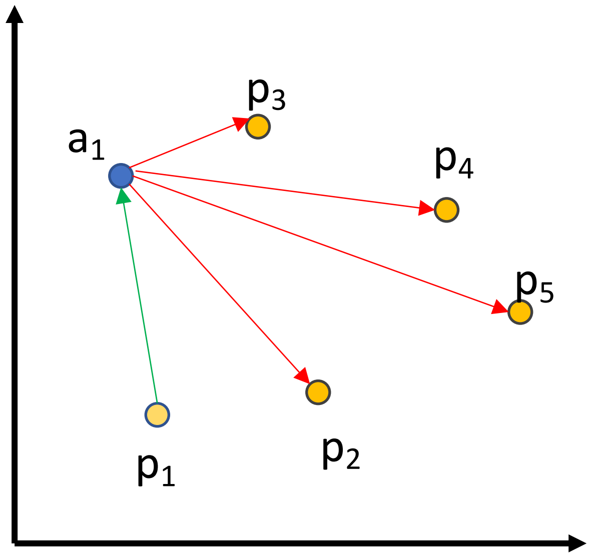
- Have positive pairs:
  (a_1, p_1)
  (a_2, p_2)
  (a_3, p_3)

- 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 != 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

Multiple Negative Ranking Loss 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:
  - $sim(a_1, p_1) = 0.5$
  - $sim(a_1, p_2) = 0.3$
  - $sim(a_1, p_3) = 0.1$

- 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:

$$\text{sim(vec\_a, vec\_b) = vec\_a * vec\_b}^{T} = \begin{bmatrix} \text{sim}(a_1, p_1), \text{sim}(a_1, p_2), \text{sim}(a_1, p_3) \\ \text{sim}(a_2, p_1), \text{sim}(a_2, p_2), \text{sim}(a_2, p_3), \\ \text{sim}(a_3, p_1), \text{sim}(a_3, p_2), \text{sim}(a_3, p_3) \end{bmatrix}$$

- 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

```
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 sim(a, b)?
  - a, b are vectors
  - Dot-product: dot_prod(a, b) = ab^T
  - Cosine-Similarity: cos_sim(a, b) = (ab^T) / (||a|| ||b||)
    - Does not work well, scores differences are too small
  - Scaled Cosine-Similarity: scaled_cos_sim(a, b) = C * cos_sim(a, b)
    - Works well with e.g. C=20
  - Scaled dot-product: scaled_dot_prod(a, b) = C * dot_prod(a, b)

Cosine-Similarity vs. Dot-Product

Cosine-Similarity

- Vector has highest similarity to itself
  - cos_sim(a, a) = 1
- With normalized vectors, equal to dot_product
  - With max vector length = 1
- With normalized vectors, proportional to Euclidian distance
  - Works with k-means clustering

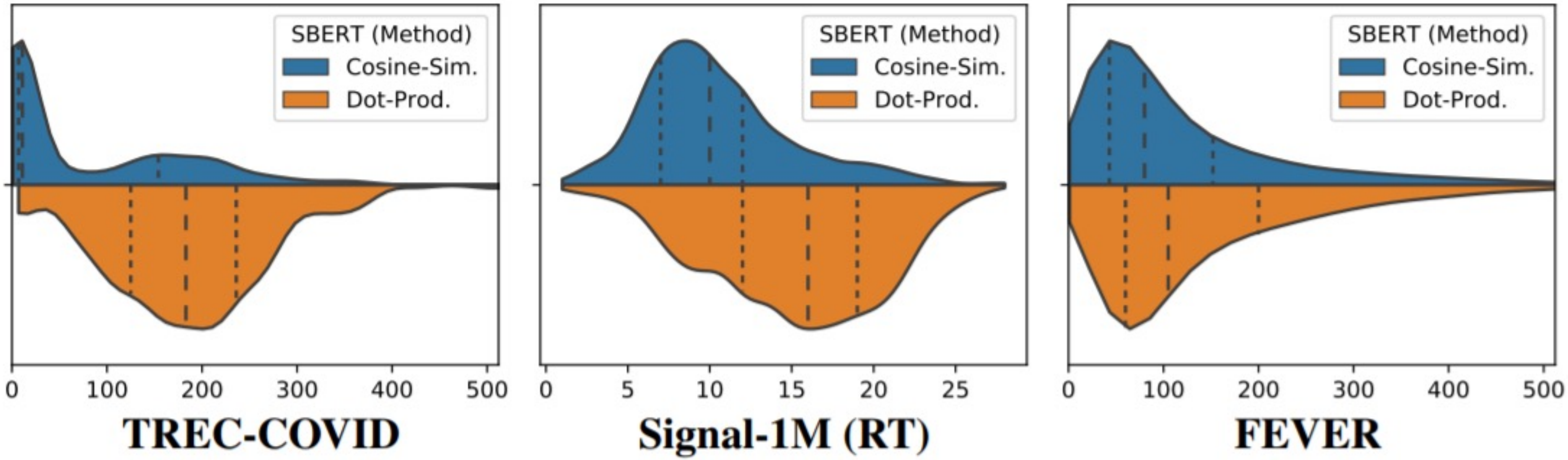
Dot-Product

- Other vectors can have higher dot-product
  - dot(a, a) < dot(a, b)
- Might be slower with certain approximate nearest neighbor methods
  - Max vector length not know
- Does not work with k-means clust.

https://arxiv.org/abs/2104.08663

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/XboxInnerProduct.pdf

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)

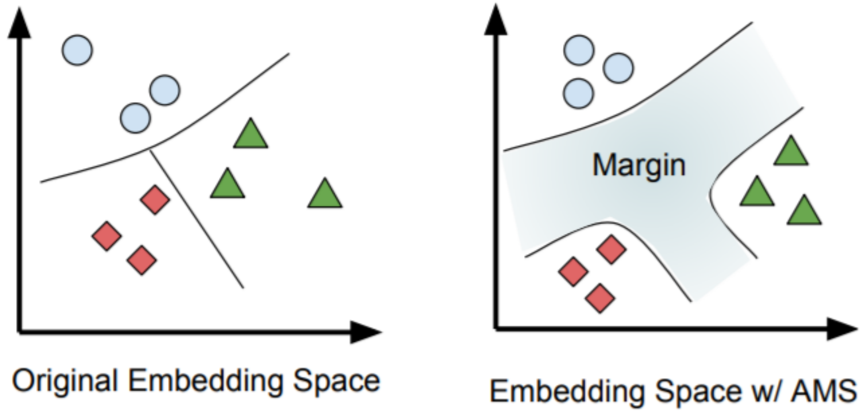
https://arxiv.org/abs/2104.08663

Optimizing the Multiple-Negatives-Ranking-Loss

- Training with scaled_cos_sim(a, b) = C * 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: scaled_cos_sim(a, b) = exp(C) * 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: (Loss(A, P) + Loss(P, A)) / 2
  - Swap anchor & positives (e.g. given answer, what is the question?
  - Unclear impact

Multiple-Negatives-Ranking-Loss with Additive Margin

- $\mathrm{sim}(\mathrm{a_i}, \mathrm{p}_j) = \begin{cases} sim(a_i, pi) - m & if\ i = j \\ sim(ai, pj) \end{cases}$



- 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

- Used in: LaBSE: https://arxiv.org/abs/2007.01852 & https://arxiv.org/abs/1902.08564

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?

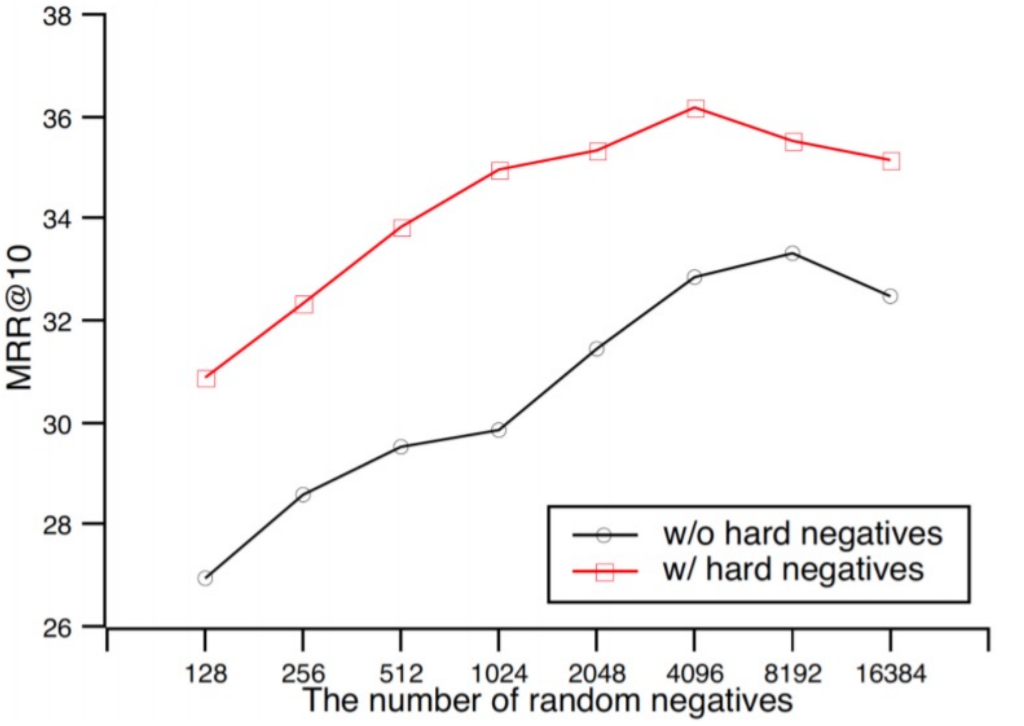
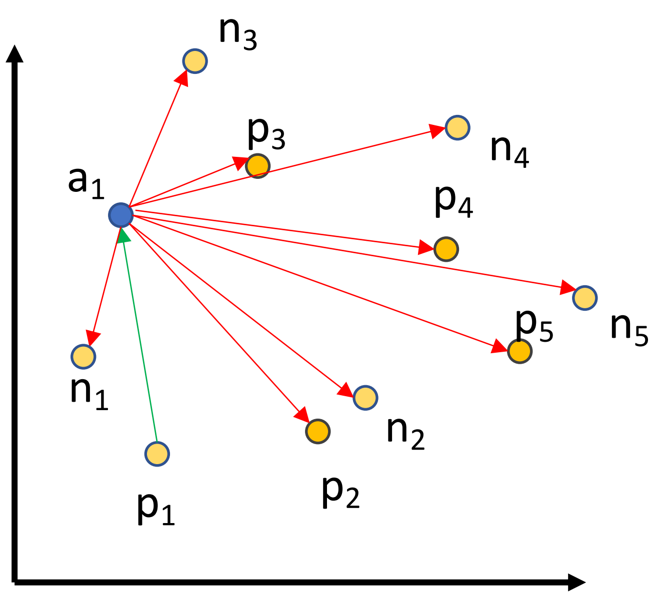


Image: https://arxiv.org/pdf/2010.08191.pdf

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…

Improving Quality with Better Batches

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

- Better approach
  - Sample pairs from one subforum (e.g. StackOverlow)

    [(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. StackOverlow, Python tag)
    [(question_python, answer_python),
      (question_numpy, answer_numpy),
      (question_pandas, answer_pandas)]

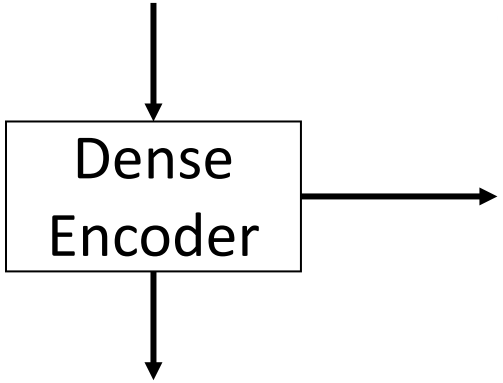
- Adding random batches might still be needed
  - Otherwise StackOverflow vector space could overlap with Travel vector space
  - 90% difficult batches, 10% easy random batches
  - Or: start with mainly random batches, then go to difficult batches

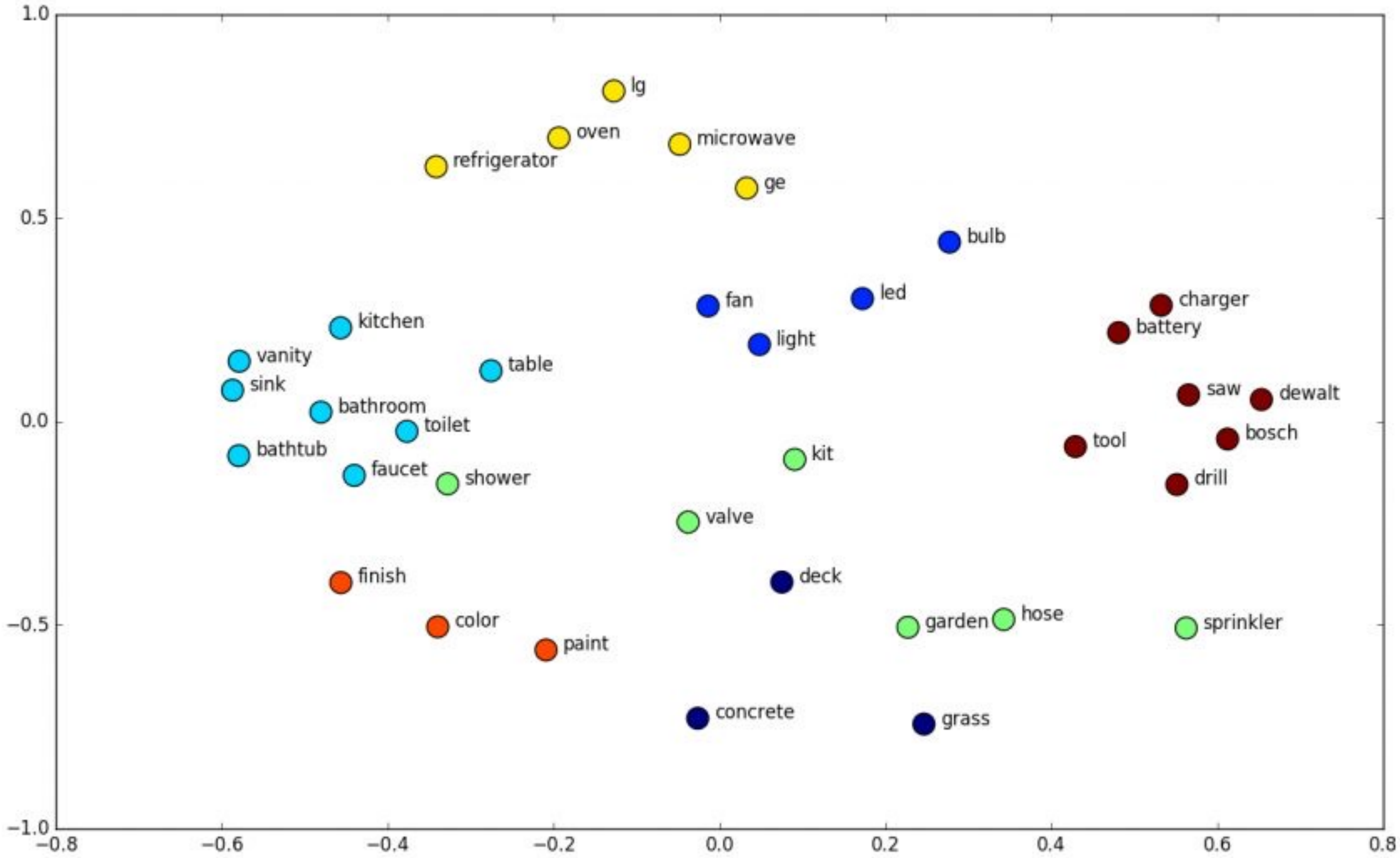
Bi-Encoders and the Curse of the Unknowns

- 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"

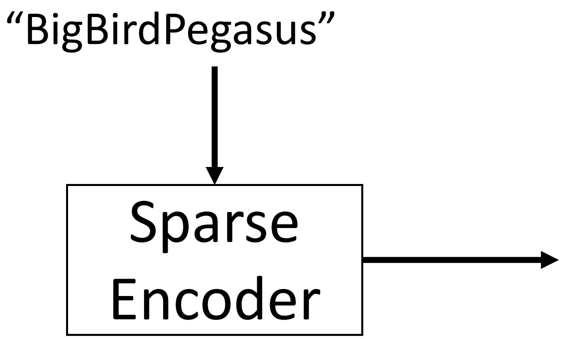




1) bigboss
2) bigdata
3) bigass

Img: https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/

Unknown Words for Sparse Bi-Encoders



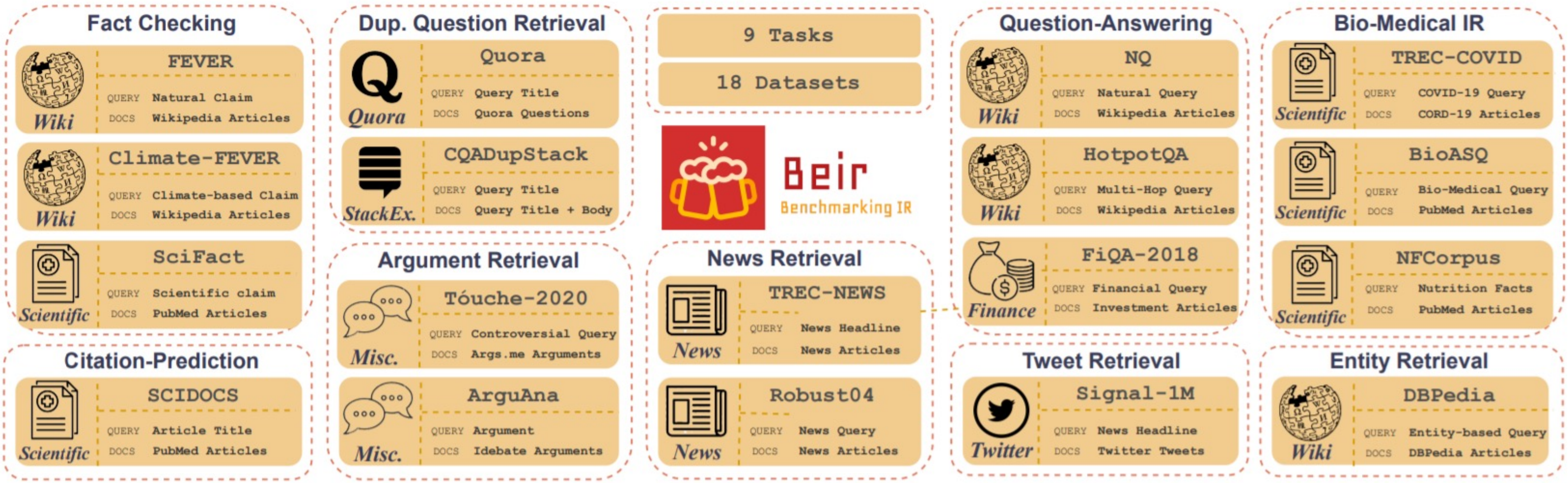
Split query in word pieces:

- 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

BEIR – Benchmarking IR



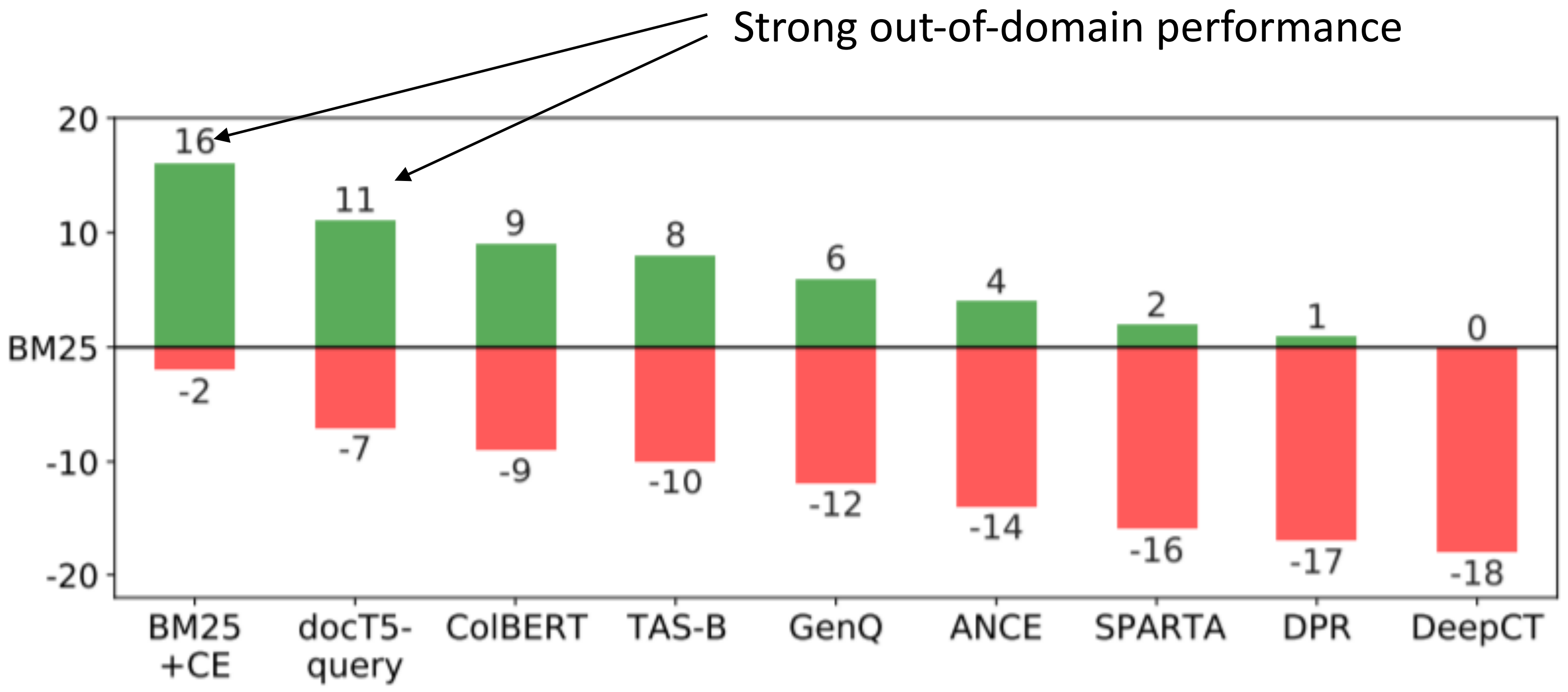
BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models, https://arxiv.org/abs/2104.08663

Bi-Encoders vs Lexical Search

Dataset	BM25	Dense Model (TAS-B)	Difference
In-Domain	22.8	40.8	+18.0
BioASQ	46.5	38.3	-8.2
SCIDOCS	15.9	14.9	-1.0

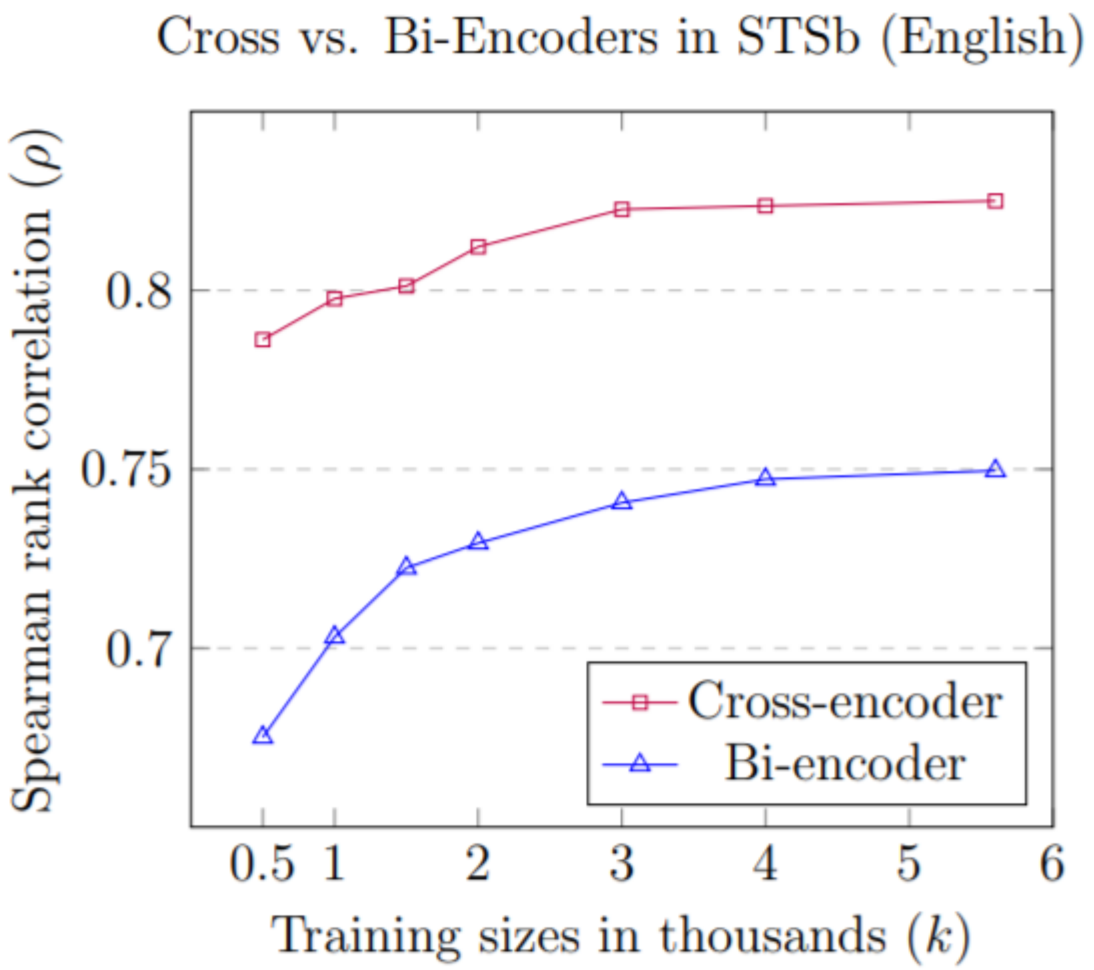
- BM25 was better on 10 / 18 datasets

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



Img: https://arxiv.org/pdf/2010.08240.pdf

Cross-Encoders vs Bi-Encoders

Dataset	BM25	Dense Model (TAS-B)	BM25 + CE
In-Domain	22.8	40.8	41.3
BioASQ	46.5	38.3	52.3
CQADupStack	29.9	31.4	37.0
TREC-COVID	65.5	48.1	75.7
SCIDOCS	15.9	14.9	16.6

- 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

Source: https://arxiv.org/pdf/2108.08513.pdf
https://medium.com/@nils_reimers/openai-gpt-3-text-embeddings-really-a-new-state-of-the-art-in-dense-text-embeddings-6571fe3ec9d9

How to Adapt Bi-Encoders to New Domains?

Adaptive Pre-Training

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

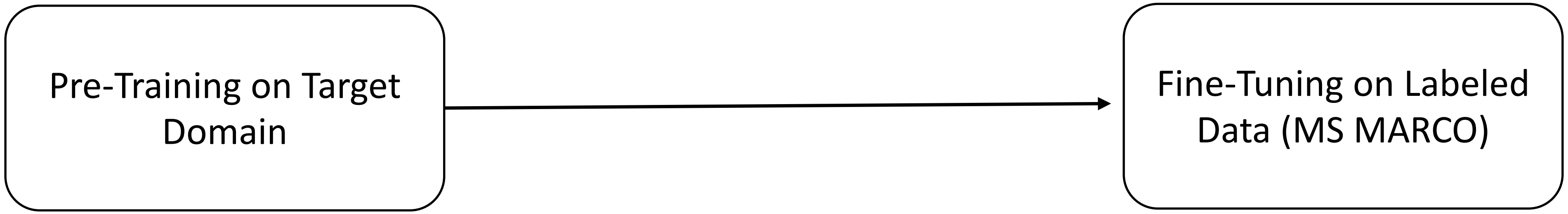
❌ Requires (expensive) training on labeled source data

What we want:

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

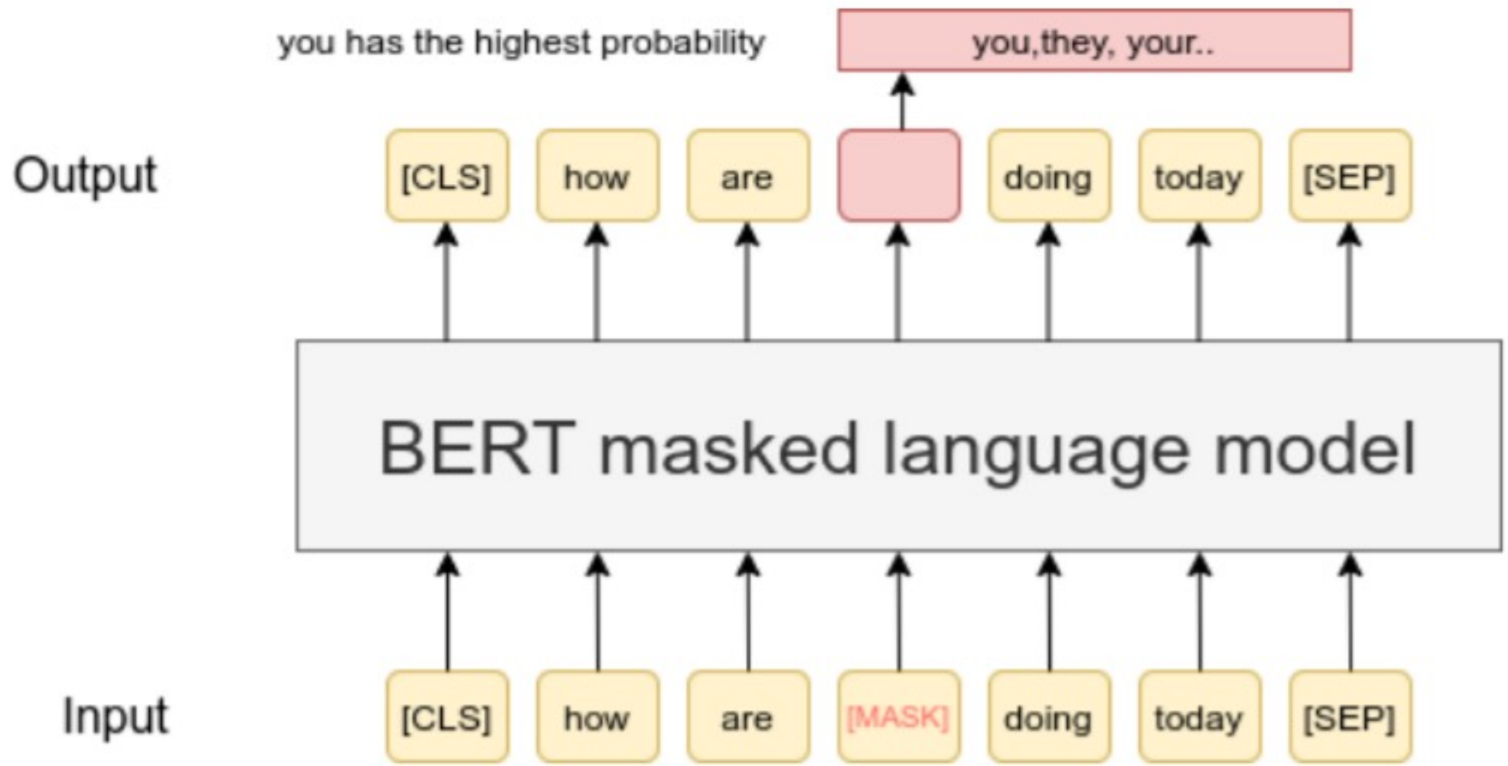
✓ Use pre-trained models and adapt to you domain

Adaptive Pre-Training



Methods for Pre-Training	**Does it work?**
Masked Language Modeling (MLM)	Yes
TSDAE	Yes
Inverse Cloze Task (ICT)	Yes
SimCSE	No – weaker than base model
Contrastive Tension (CT)	No – weaker than base model
Condenser (CD)	No – weaker than base model

Masked Language Model (MLM)



Inverse Cloze Task (ICT)

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

Select sentence at random

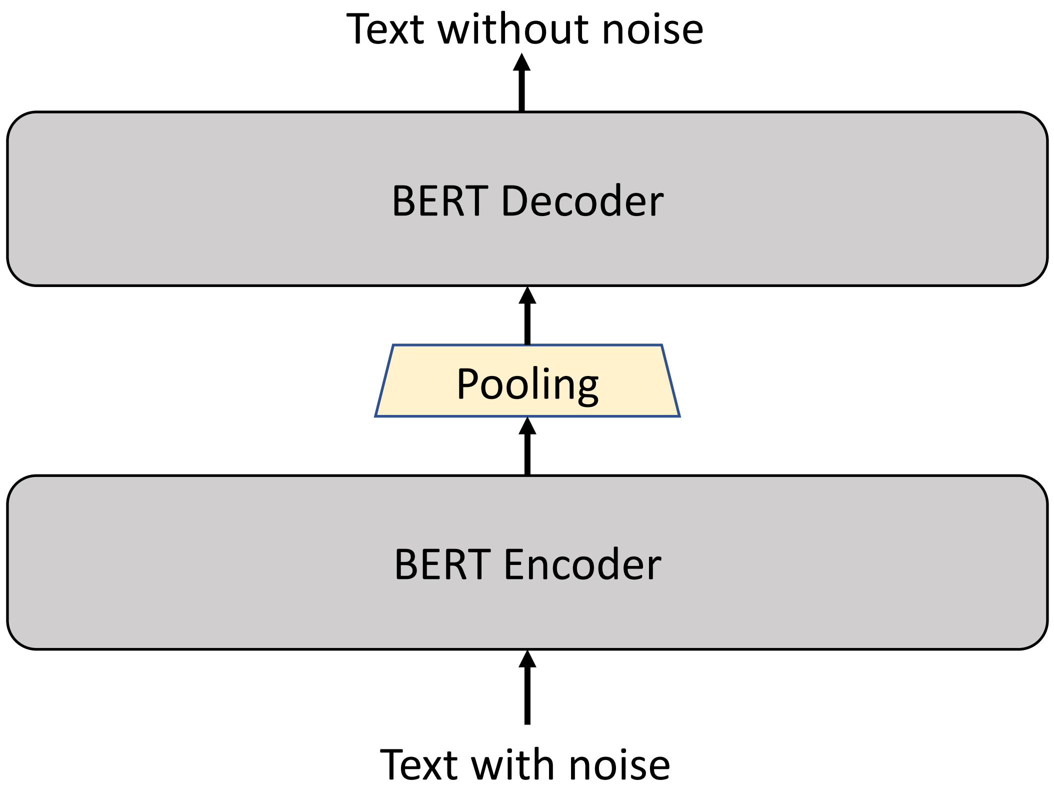
Its design philosophy emphasizes code readability.

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

Models	4 Sentence Tasks	6 Dense IR Tasks
Out-of-the-box	52.3	45.2
Source -> Target		
TSDAE	54.2	-
MLM	51.1	-
Target -> Source		
TSDAE	56.5	49.2
MLM	55.9	46.7
ICT	-	46.5
SimCSE	52.4	45.0
CD	-	44.7
CT	53.0	44.0

https://arxiv.org/abs/2104.06979 / https://arxiv.org/abs/2112.07577

Domain Adaptation on Pre-Trained Model

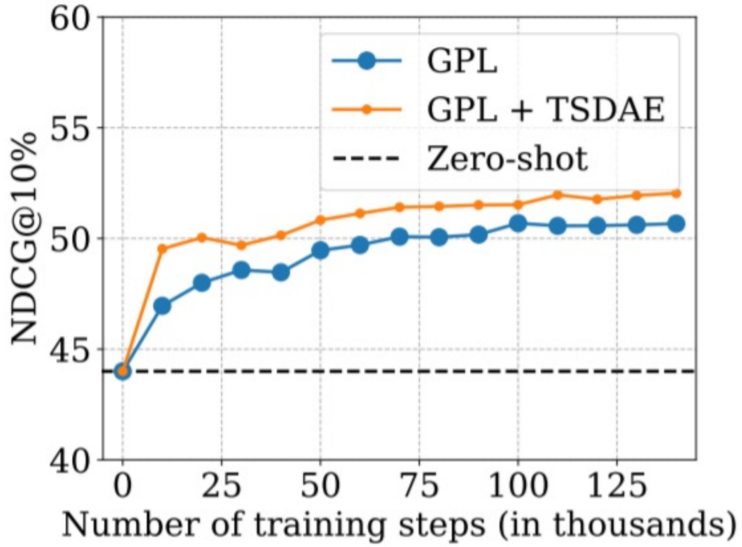
❌ Adaptive pre-training is expensive

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

✔ What we want:

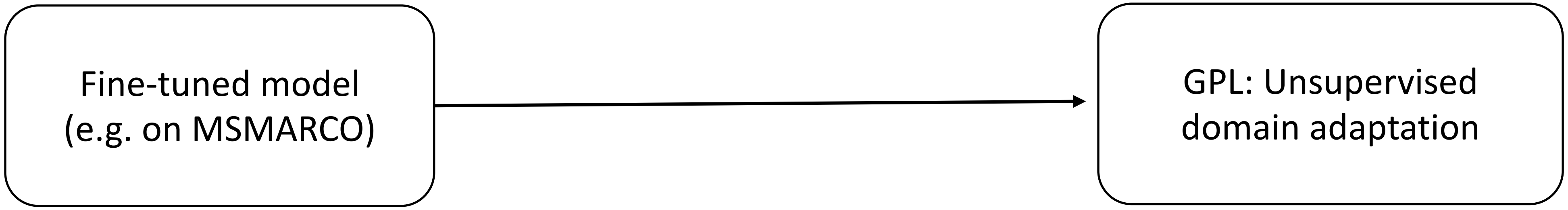
1) Fine-tuning on labeled source dataset (can be as large as 1B+ training pairs)
2) Unsupervised training on target domain

Generative Pseudo Labels (GPL) is able to achieve this

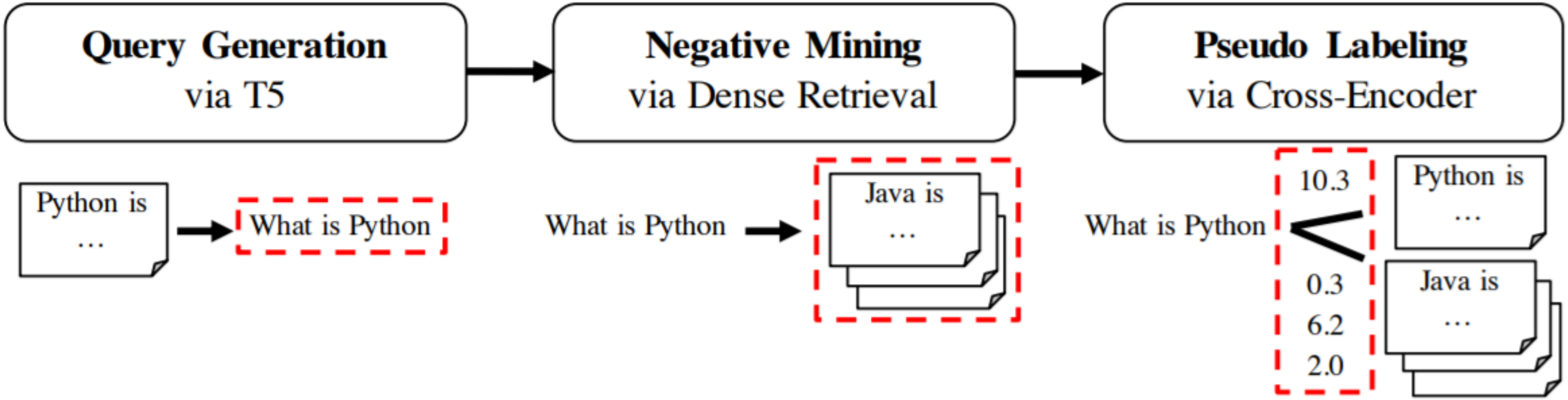


https://arxiv.org/abs/2112.07577

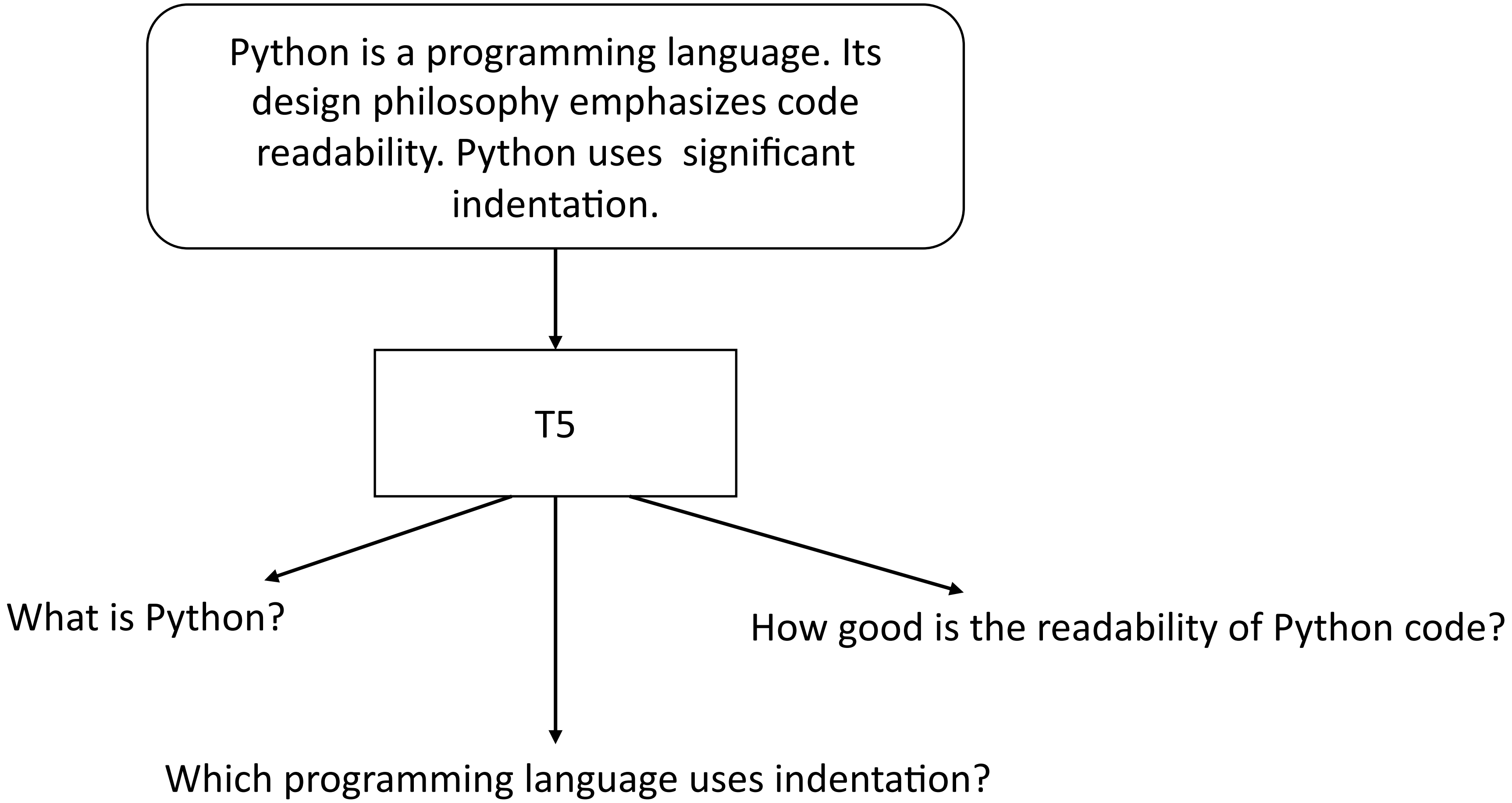
GPL – Generative Pseudo Labeling



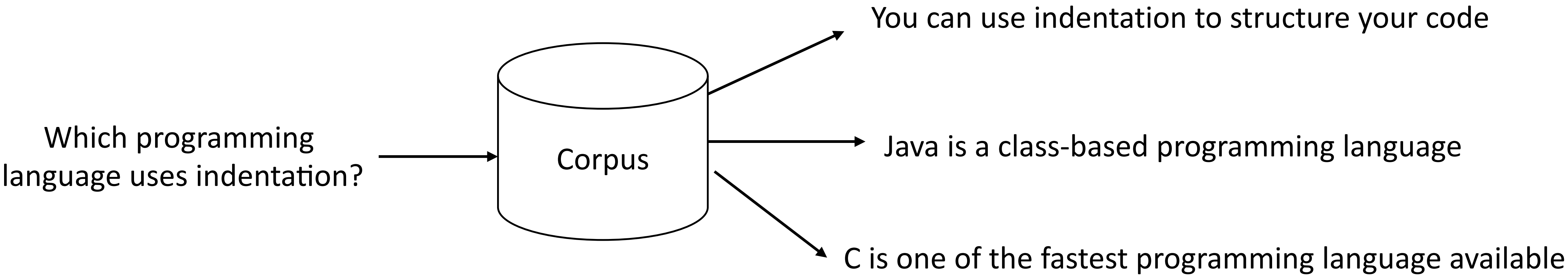
GPL:



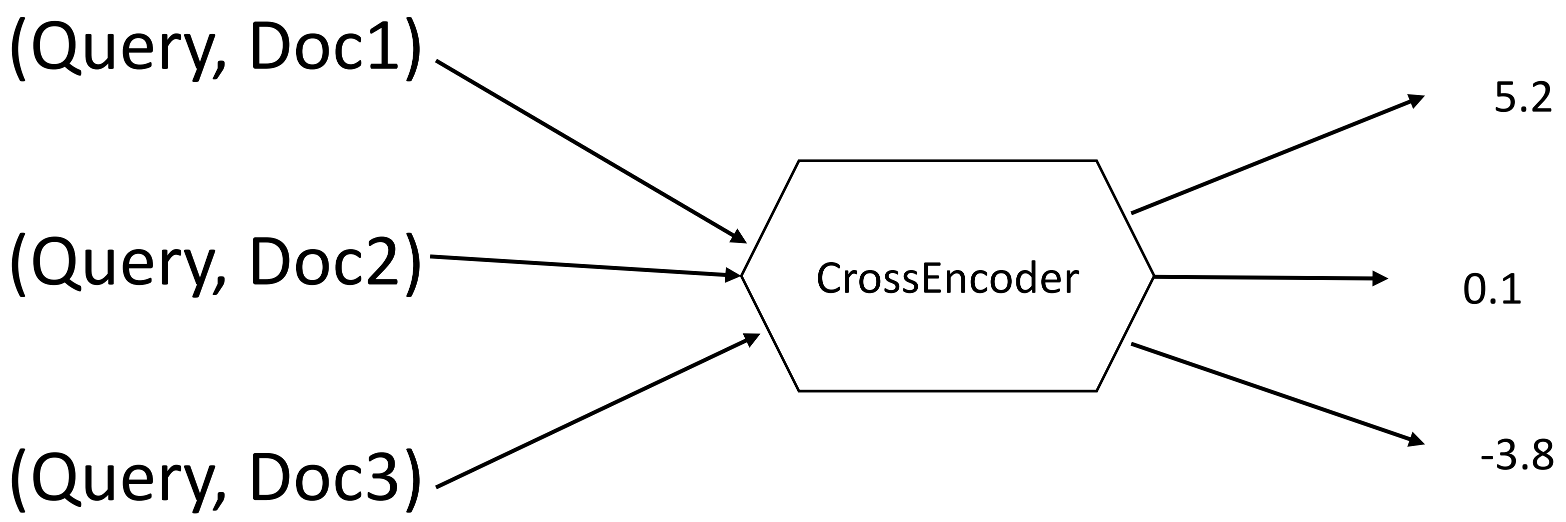
Step 1: Generate Queries



Step 2: Mine Negatives



Step 3: Score Pairs with CrossEncoder



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

Item	Text	GPL	QGen
Query	what is **futures contract**	–	–
Positive	**Futures contracts** are a member of a larger class of financial assets called derivatives ...	10.3	1
Negative 1	... Anyway in this one example the s&p 500 **futures contract** has an "initial margin" of $19,250, meaning ...	2.0	0
Negative 2	... but the moment you exercise you must have $5,940 in a margin account to actually use the **futures contract** ...	0.3	0
Negative 3	... 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...	8.2	0
Negative 4	... A **futures contract** commits two parties to a buy/sell of the underlying securities, but ...	6.9	0

Train Bi-Encoder with MarginMSE-Loss

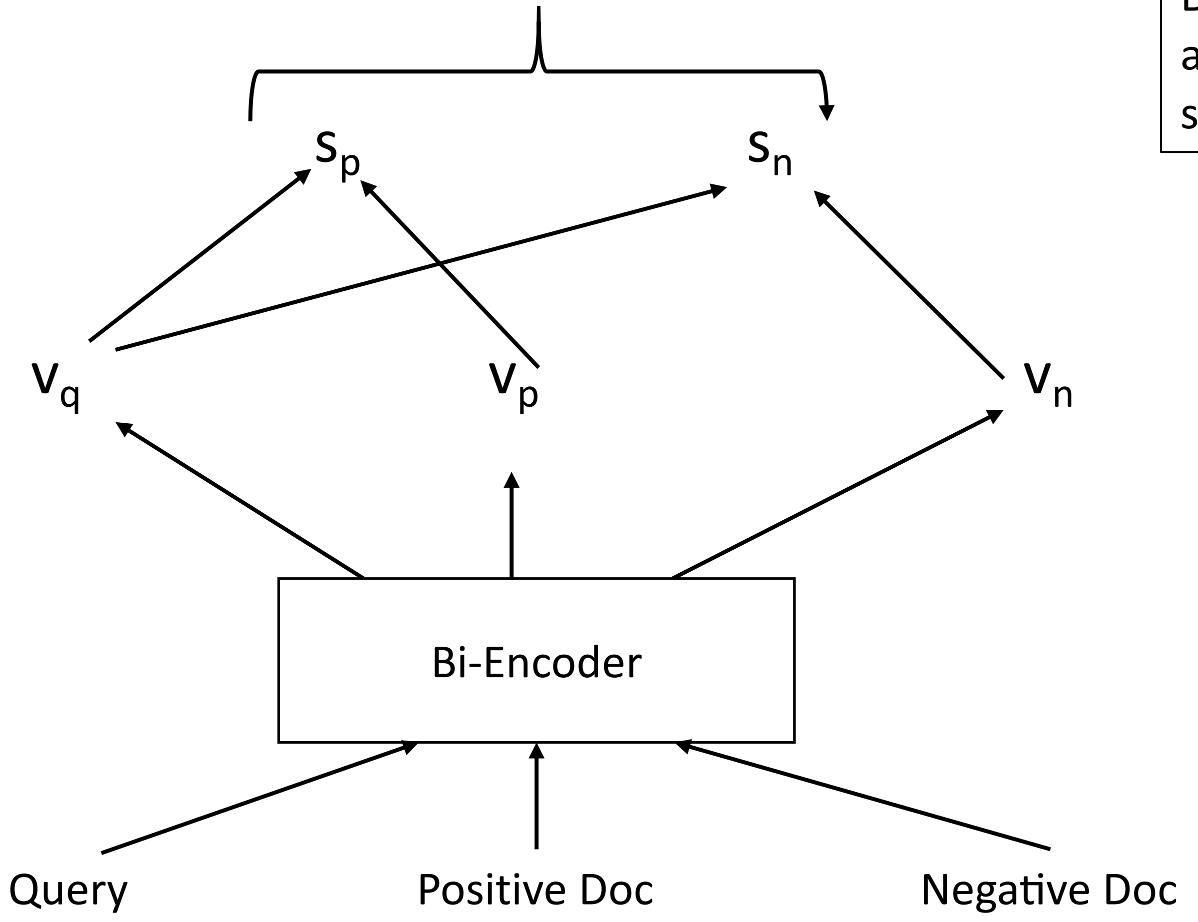
Compute Loss

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

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

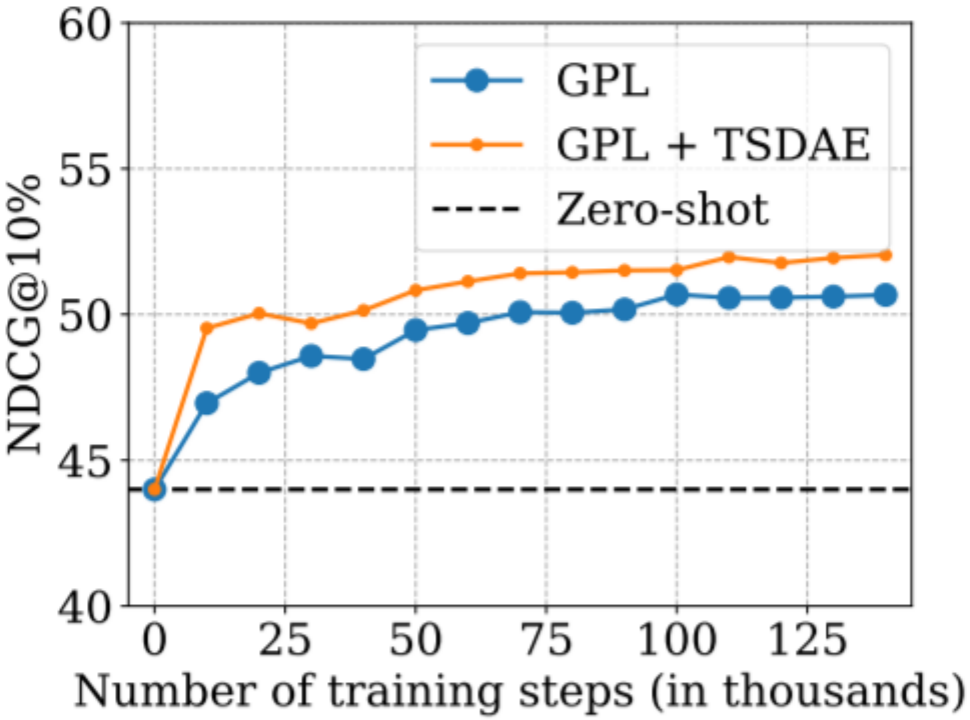
Compute dot-scores

Compute Embeddings



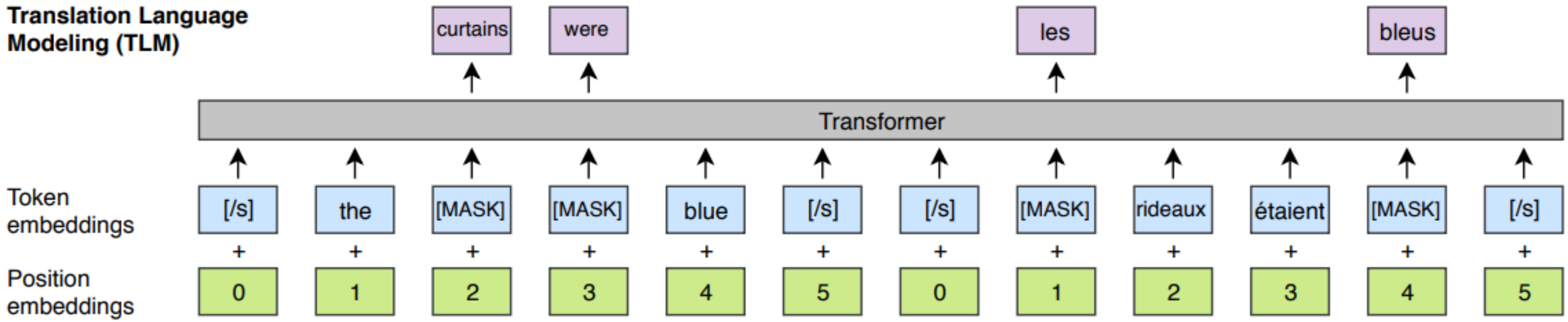
Results

Models	6 Dense IR Tasks
Out-of-the-box	45.2
Target -> Source	
TSDAE	49.2
MLM	46.7
Generative Pseudo Labeling	
GPL	51.4
TSDAE+GPL	52.4



Multilingual Models

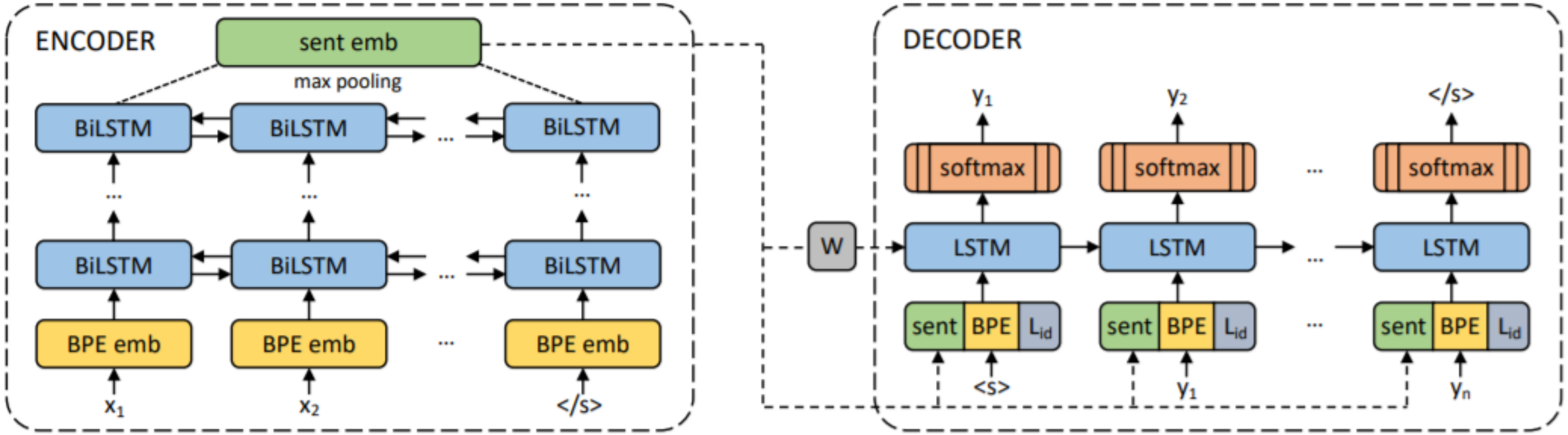
Translation Language Model



- Concatenate parallel data and run MLM

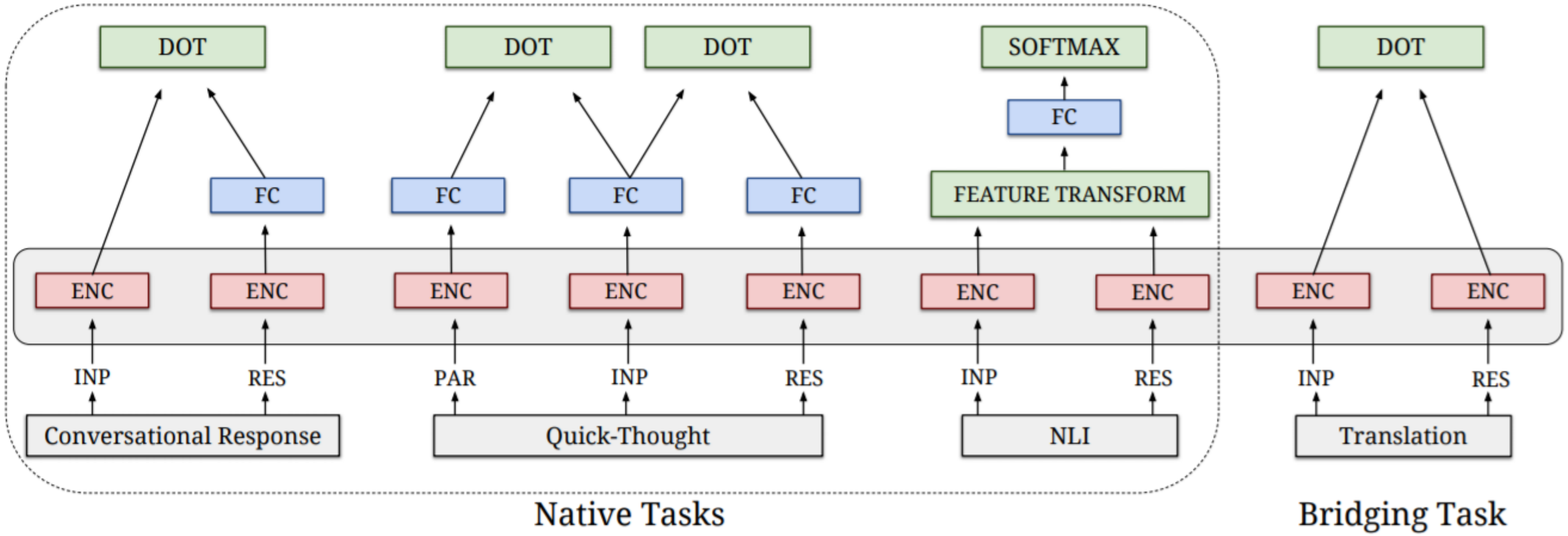
https://arxiv.org/pdf/1901.07291.pdf

Previous Approaches: LASER



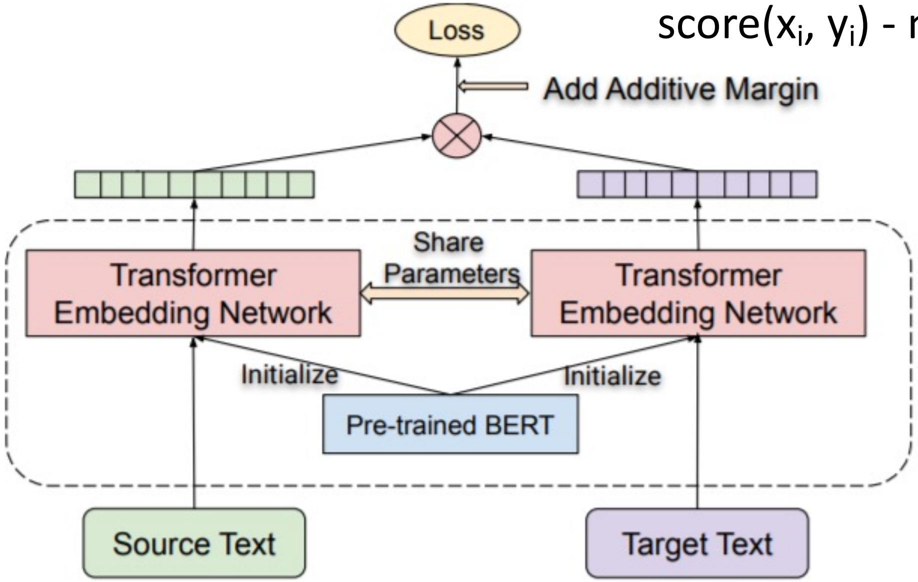
- Use output of encoder from translation system
- Issues:
  - Cannot control what type of embeddings are learned
  - Works poorly on identifying similar sentences

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



$\text{score}(x_i, y_i) - m$

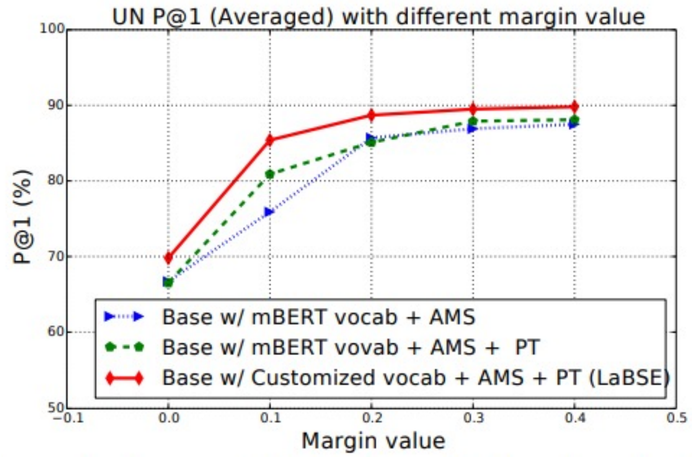
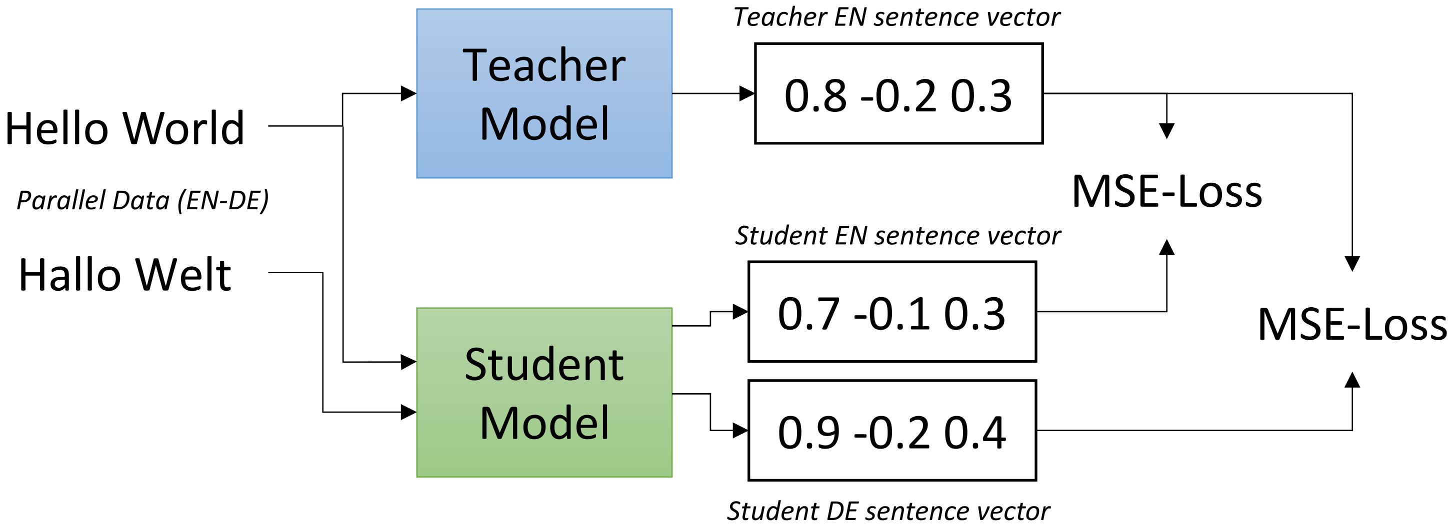


Figure 4: Average P@1 (%) on UN retrieval task of models trained with different margin values.

- 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

https://arxiv.org/abs/2007.01852

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:

$$S(s_i) \approx T(s_i) \qquad S(t_i) \approx T(s_i)$$

https://arxiv.org/abs/2004.09813

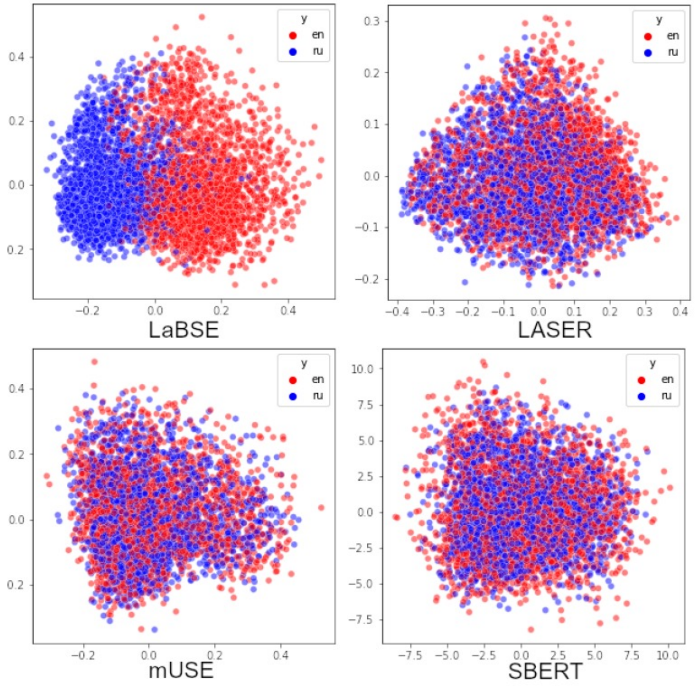
Performance

Model	STS	BUCC Bitext Mining
Knowledge Distillation	**83.7**	88.6
mUSE	81.1	87.7
LaBSE	73.5	**93.5**
LASER	67.0	93.0

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

https://arxiv.org/abs/2004.09813

Language Bias



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

Model	Expected Score	Actual Score	Difference
LASER	69.5	68.6	-0.92
mUSE	81.7	81.6	-0.19
LaBSE	74.4	73.1	-1.29
XLM-R ← SBERT-paraphrases	84.0	83.9	-0.11

https://arxiv.org/abs/2004.09813

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

शादी (hindi: wedding)

Who is the president?

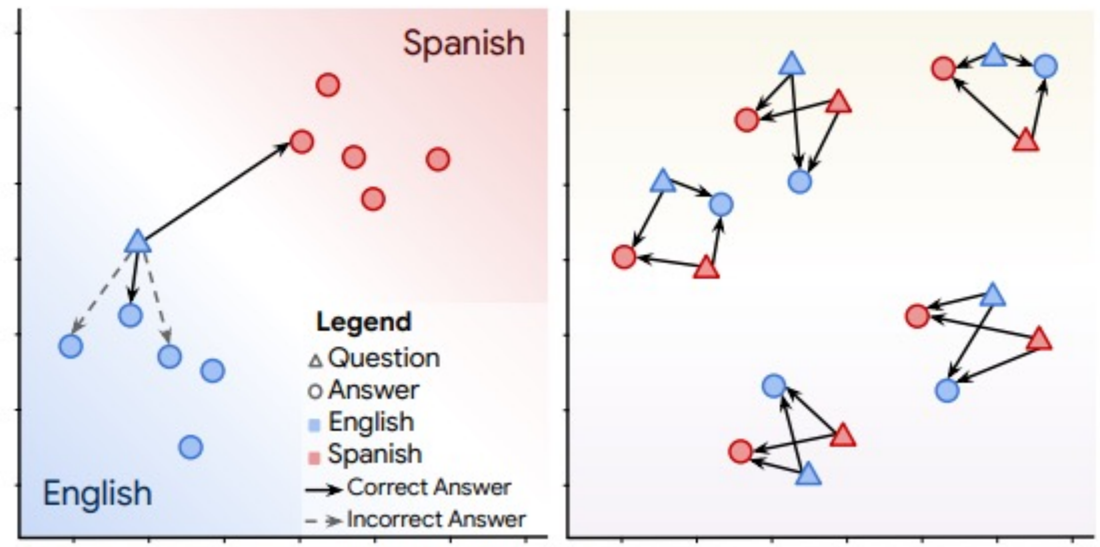
A: Joe Biden is the current president

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



(a) **Weak Alignment** (b) **Strong Alignment**

LAReQA https://arxiv.org/abs/2004.05484

Batch Strategy

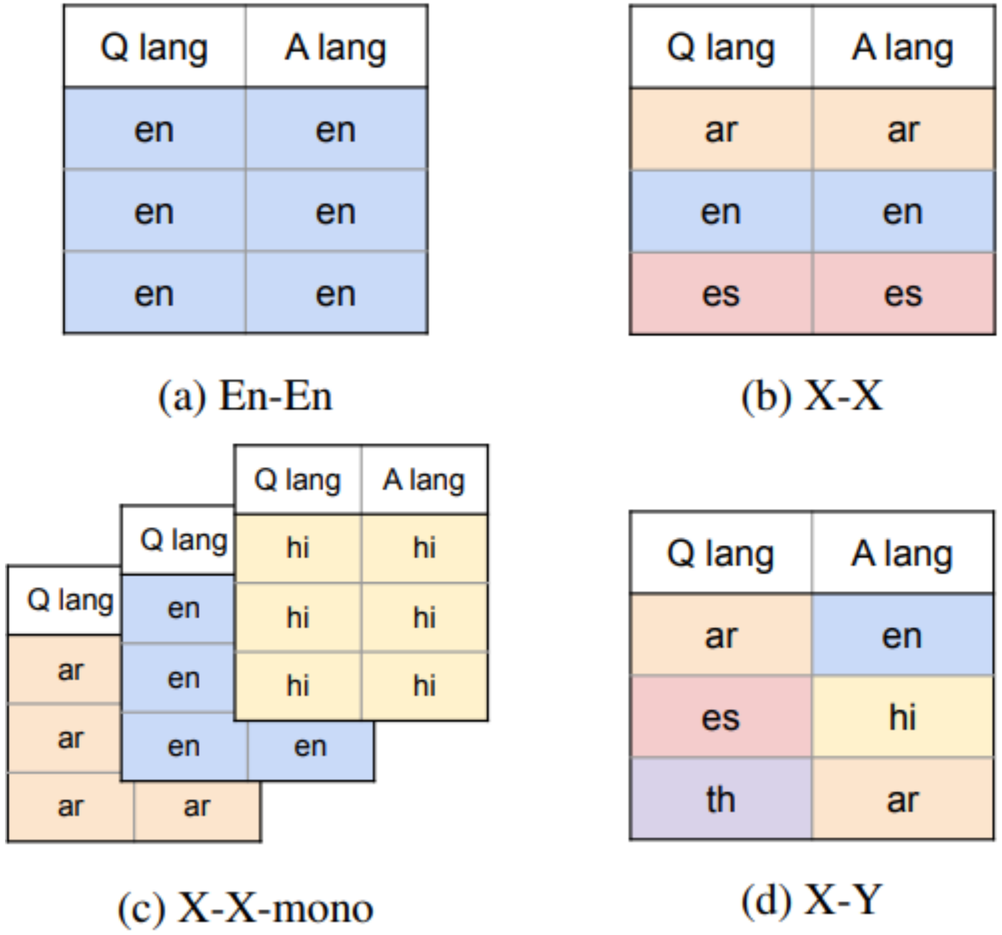


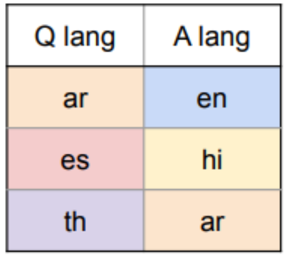
Figure 3: Sample batches for each baseline.

- 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

Conclusion

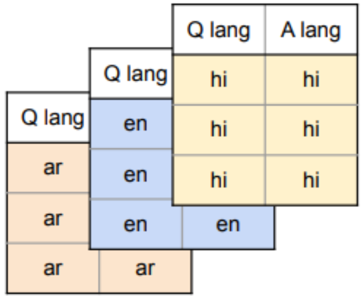
- Training for bitext mining models:
  - LaBSE

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



(d) X-Y

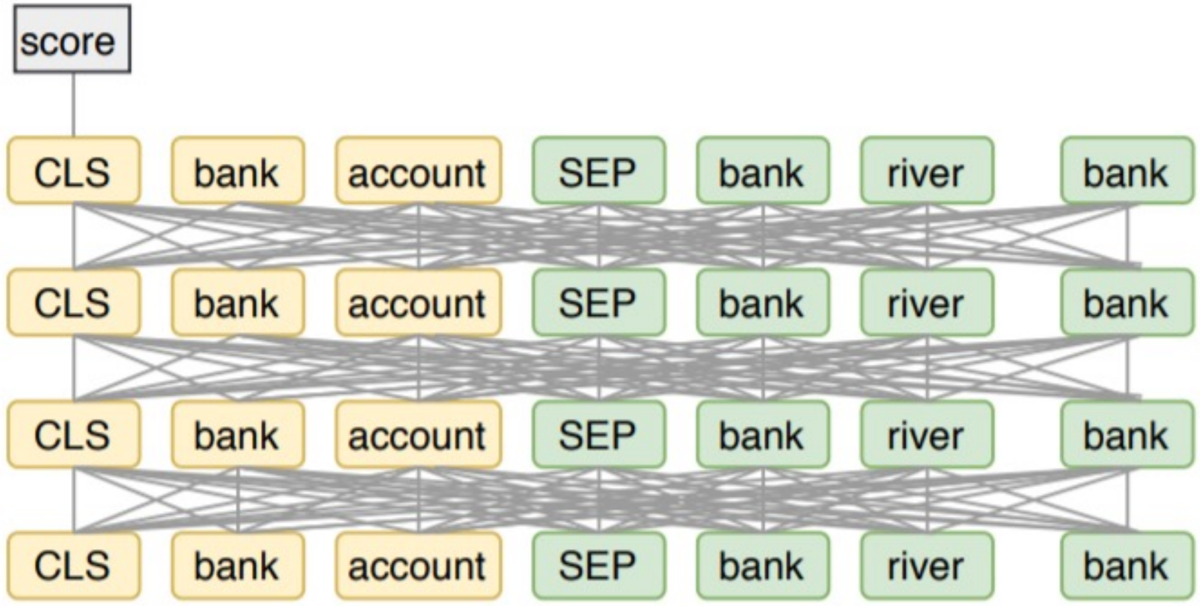
- Training for **multilingual search** models
  - X-X-mono batch strategy
  - Train on (multilingual) pair & triplet datasets
  - Add parallel data for alignment



(c) X-X-mono

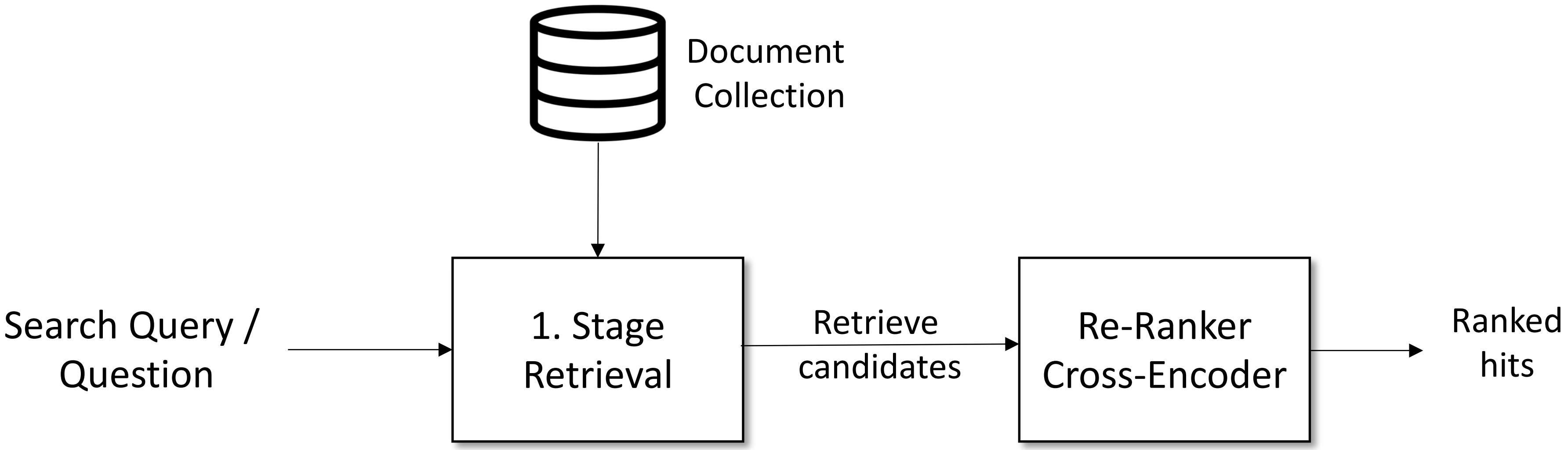
Cross-Encoder

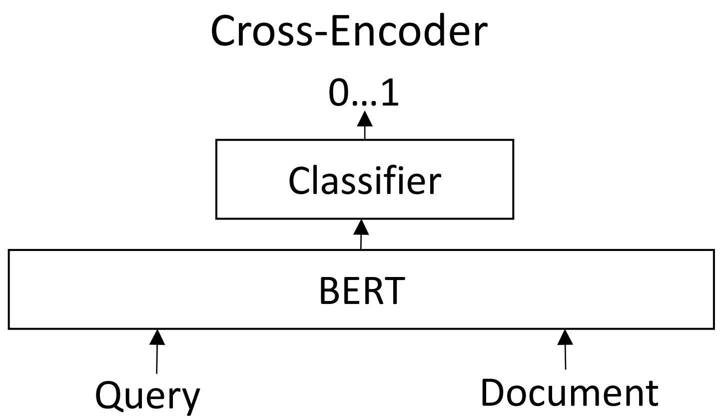
Cross-Encoder



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

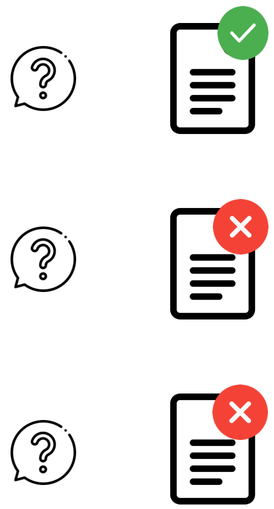
Cross-Encoder





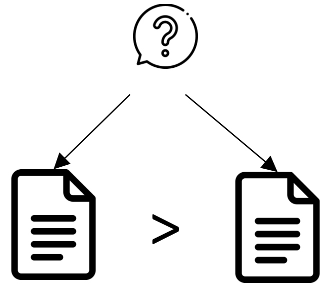
Learning to Rank – Pointwise Loss

- Pointwise-Loss
  - Given (query, document, label) triplets
  - Set label=0 / label=1 for non- / relevant docs
  - Binary classification task: BCELoss(CE(Query, Doc), 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(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, doc_1, doc_2, doc_3, …)
- 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 s_1 = CE(query, doc_1), s_2 = CE(query, doc_2), s_3 = CE(query, doc_3), …
  - 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

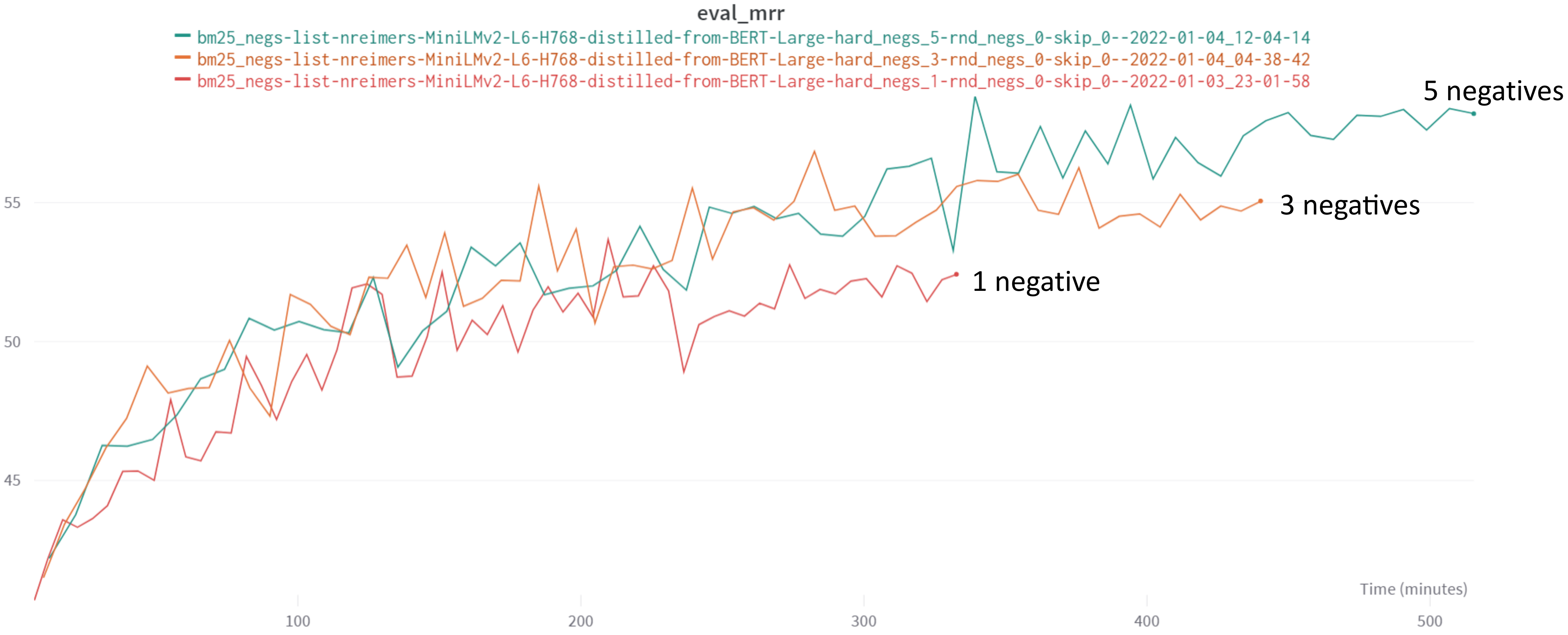
Importance of Negatives

Train Neg↓ / Inference→	**BM25**	**BM25***	**HDCT**
BM25	**39.6**	39.5	38.1
BM25*	40.7	**42.3**	41.8
HDCT	40.8	41.9	**43.4**

- 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)

https://arxiv.org/pdf/2101.08751.pdf

Number of Negatives



own experiments / https://arxiv.org/pdf/2101.08751.pdf

Multilingual Cross-Encoder

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

Model	**Performance**
mdeberta v3 (training: English only)	52.2
mdeberta v3 (14 mMARCO langs)	53.0
LaBSE	52.6
mMiniLM	52.0

- Models perform surprisingly well even when train on English only