Introduction
Dense Text Representations

Nils Reimers

Author of Sentence Transformers

www.SBERT.net

Content

- **Part 1: Introduction**
  - Background
  - Applications
- Part 2: Basic Training Methods
- Part 3: Advanced Training Methods

Why Dense Representations?

- Text = Sentence, Paragraph, Document
- Traditionally: Sparse, Lexical Representation (Bag-of-words)
  - Each word has it own dimension

How are you?

[0, 0, 1, 0, 0, 0, 1, 0, 1, …]

How are you

- Issues:
  - Lexical gap: US, USA, United States
  - Ambigue words
  - Same distance between all words
  - Word order not preserved

Dense Representation

- Formally: $f(Text) \rightarrow \mathbb{R}^n$
- *n* dimensional representation (embedding)
- Find function f such that semantically similar text is close

What does semantically similar mean?

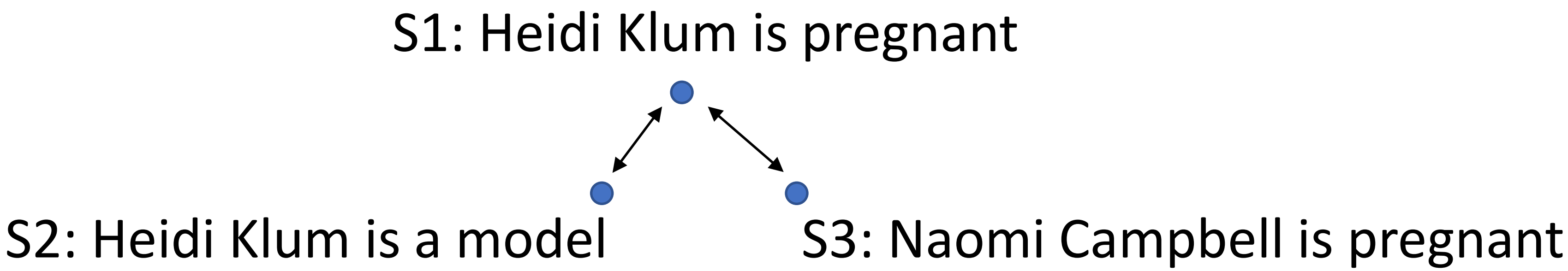
- Find function *f* such that **semantically similar** text is close
- There cannot be ***universal*** *text representations*

Nuclear energy is safe! ⟷ Nuclear energy is dangerous!

Semantically similar depends on the task!

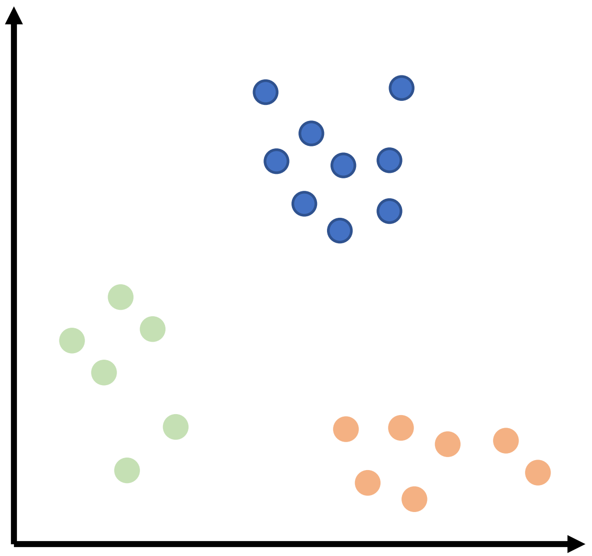
What does semantically similar mean?

- Find function f such that **semantically similar** text is close
- There cannot be ***universal** text representations*



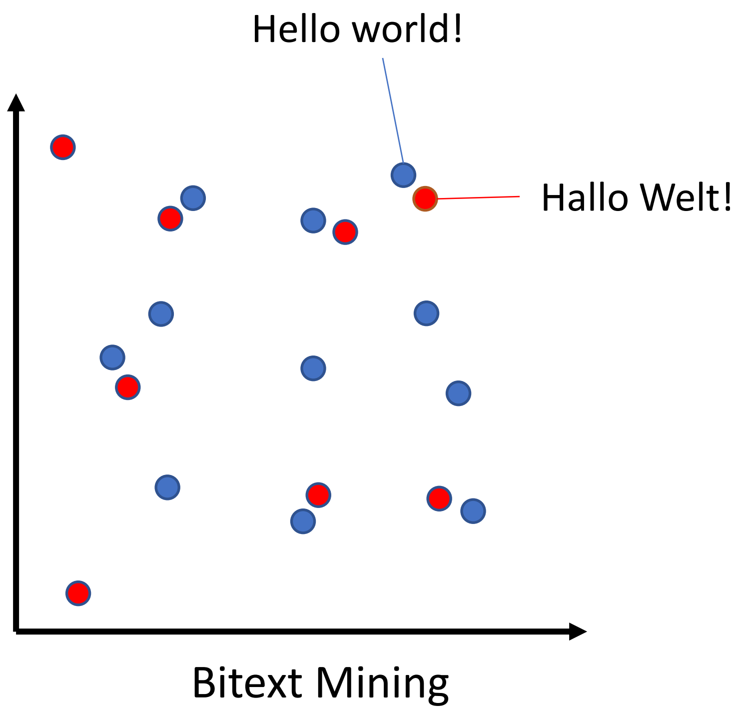
Semantically similar depends on the task!

Application - Clustering

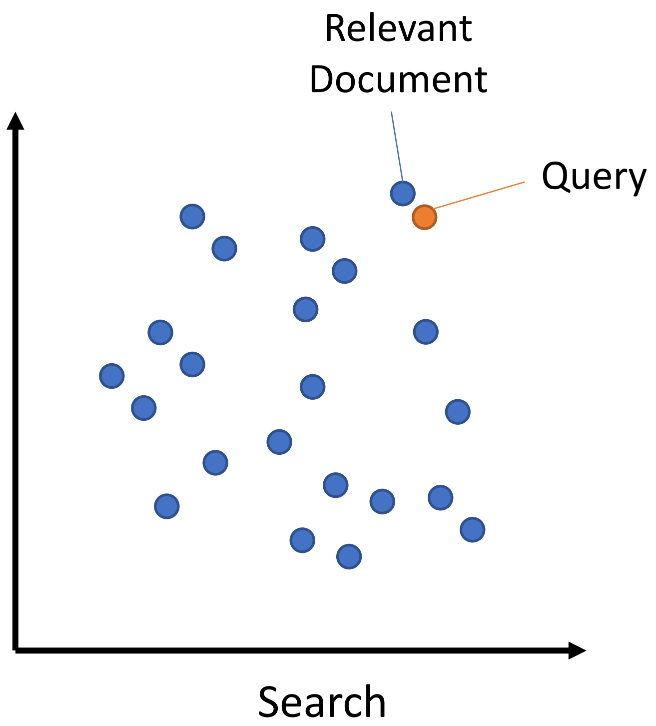


Clustering

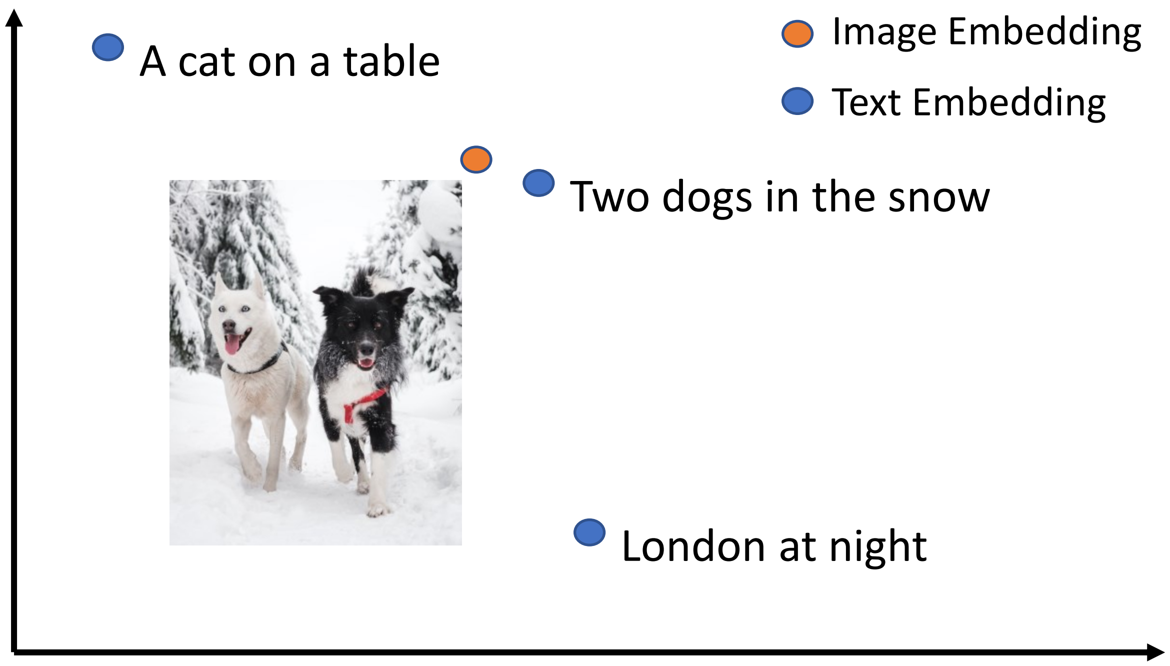
Application – Bitext Mining

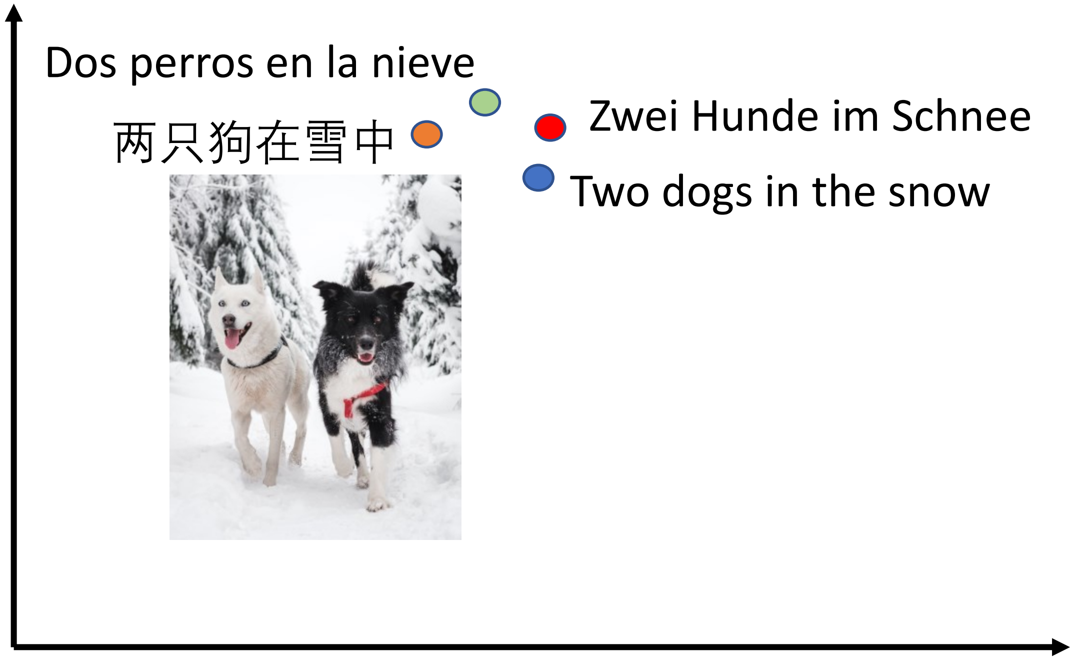


Application – Search



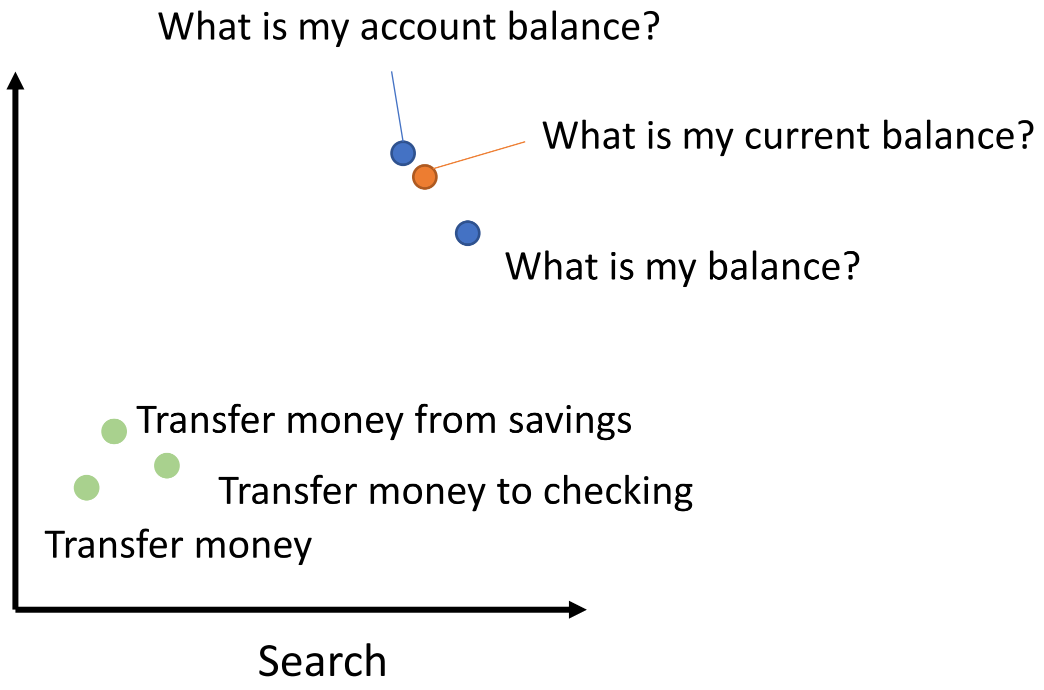
Multi-Modal Search





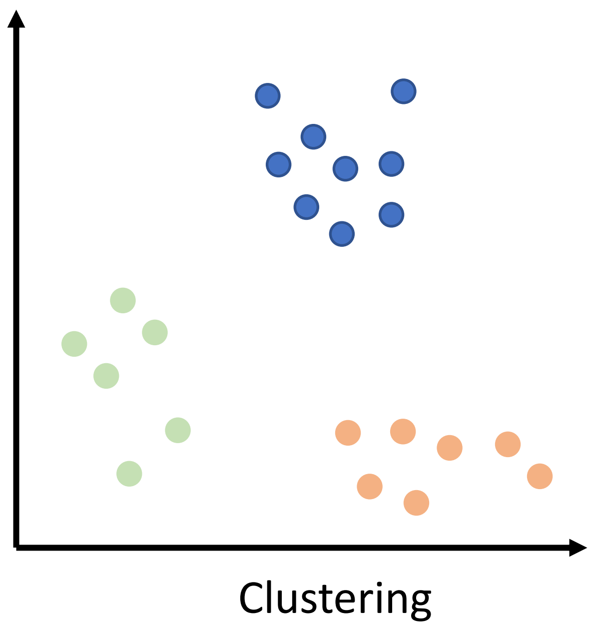
Zero-Shot Image Classification

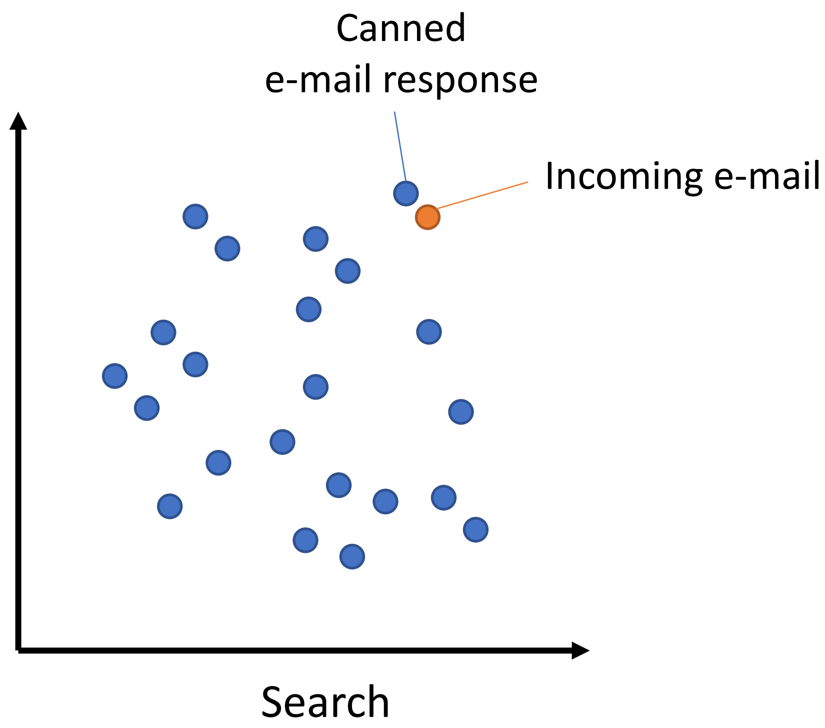
Few-Shot Intent Classification



- Have some examples for every intent (checking balance, transfer money)
- New utterance => find closest example => use intent

Application – Automate E-Mail Support





- Find most common e-mails in your inbox
- Create canned responses for top 100 questions
- Train model on (email, response)

- For new email: What is the closest (canned) response?

Conclusion

- Map text/images/… to low dimensional dense vector space
- Semantically similar text should be close
  - No clear definition what “semantically similar” means => depends on task
- Many applications. Most promising:
  - Search
  - Mining – find related items

Basic Training
Dense Text Representations

Nils Reimers

Author of Sentence Transformers

www.SBERT.net

Content

- Part 1: Introduction
- **Part 2: Basic Training Methods**
  - Basic training method
  - Loss functions
  - Improving training quality with hard negatives
- Part 3: Advanced Training Methods

Average Word Embeddings

- Simple baseline: Average word embeddings in a sentence

$$\overset{W_1}{\begin{bmatrix} W_{11} \\ W_{12} \\ \\ \\ W_{1n} \end{bmatrix}} + \overset{W_2}{\begin{bmatrix} W_{21} \\ W_{22} \\ \\ \\ W_{2n} \end{bmatrix}} + \; \cdots\cdots \; + \overset{W_n}{\begin{bmatrix} W_{n1} \\ W_{n2} \\ \\ \\ W_{nn} \end{bmatrix}} = \overset{D}{\begin{bmatrix} \dfrac{W_{11}+W_{21}+\ldots+W_{n1}}{n} \\ \\ \\ \dfrac{W_{1n}+W_{2n}+\ldots+W_{nn}}{n} \end{bmatrix}}$$

- Improvement: Weight word embeddings by tf-idf
  - Content words contribute more than stop words
- Fast and simple method
- Can be used with custom trained word embeddings

Source: http://yaronvazana.com/2018/09/20/average-word-vectors-generate-document-paragraph-sentence-embeddings/

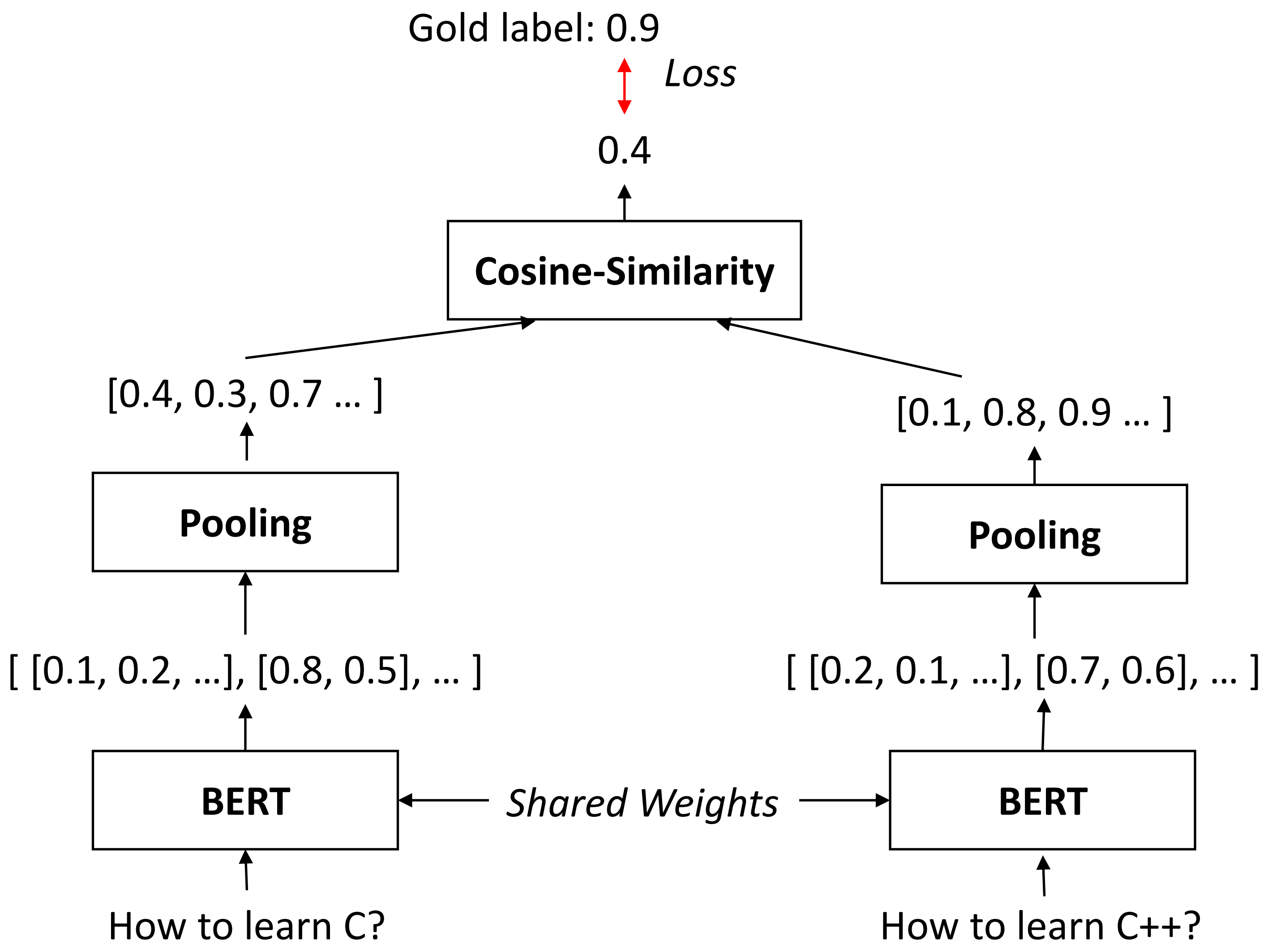
Can we just average BERT?
=> No!

Task	Avg. performance on 14 datasets
bert-base-uncased	48.5
Avg. word embeddings	51.1
Best unsupervised method based on BERT-large	60.8
SBERT (NLI data)	62.5
SBERT (large train data)	68.0

- BERT out-of-the-box performs badly
- Labeled / structured data important

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

Sentence BERT – Cosine Similarity Loss



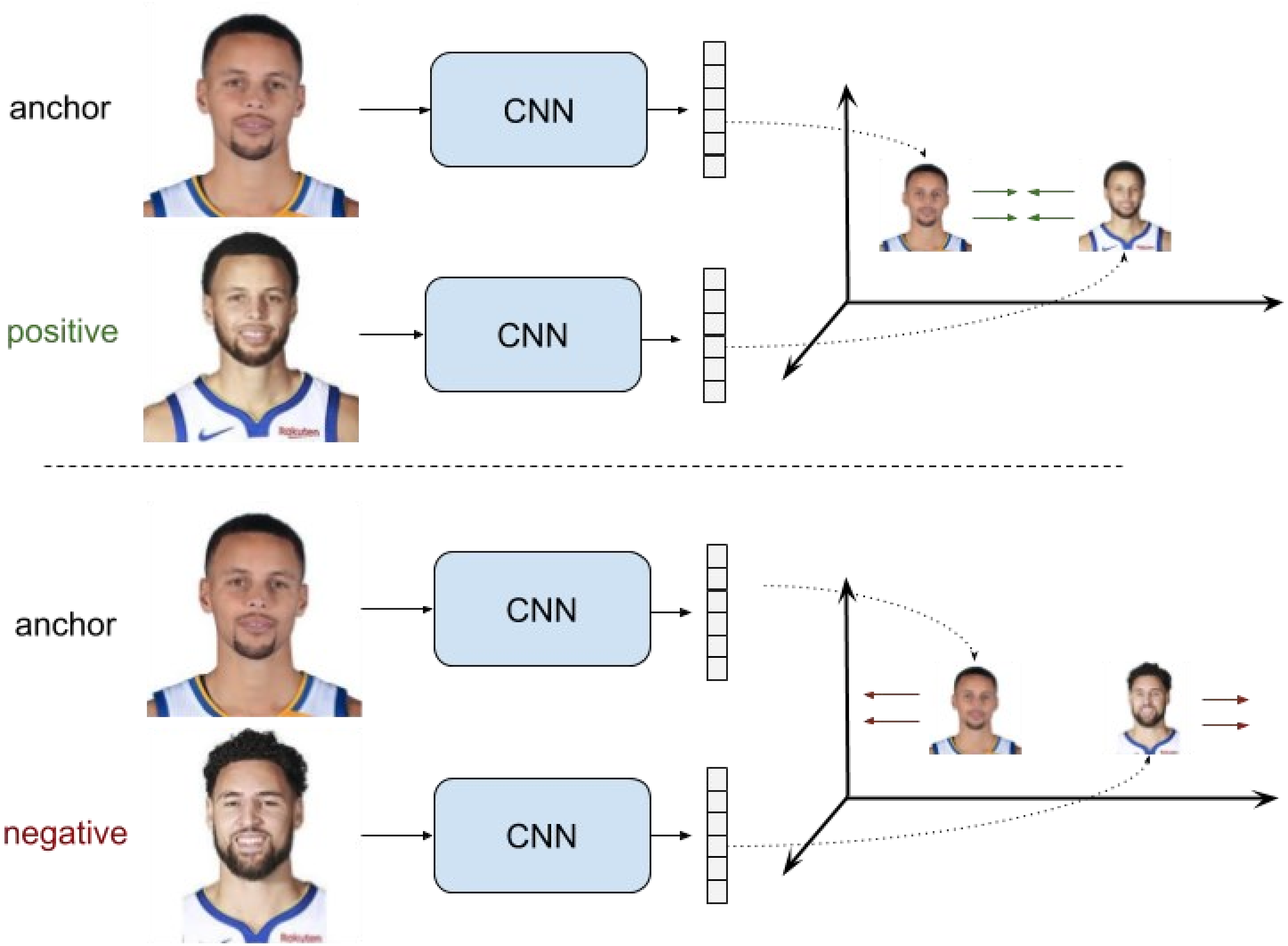
Nils Reimers, Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. EMNLP 2019

Loss Functions

- Various loss functions available to train vector space models:
  - CosineSimilarityLoss
  - SoftmaxLoss
  - Constrative- & OnlineConstrativeLoss
  - TripletLoss
  - Batch-[All | Hard | SemiHard | HardSoftMargin]-TripletLoss
  - MegaBatchMarginLoss
  - MultipleNegativesRankingLoss
  - ….

Constrative Loss

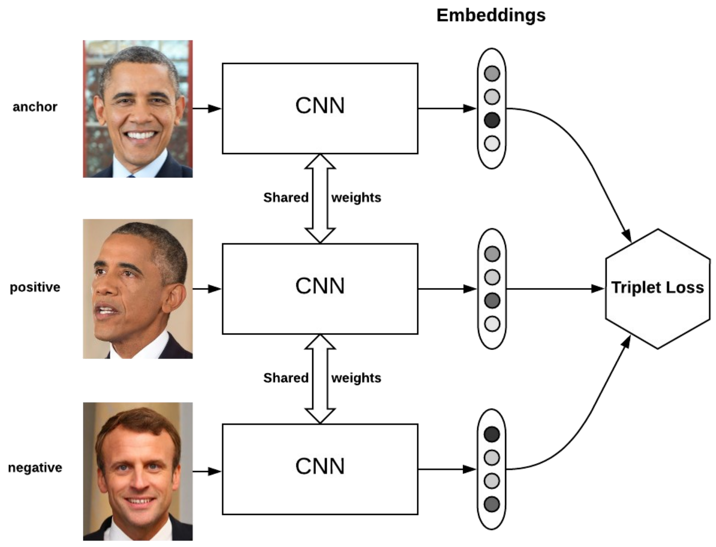
- Positive pairs
  - Pull together
- Negative pairs
  - Push away

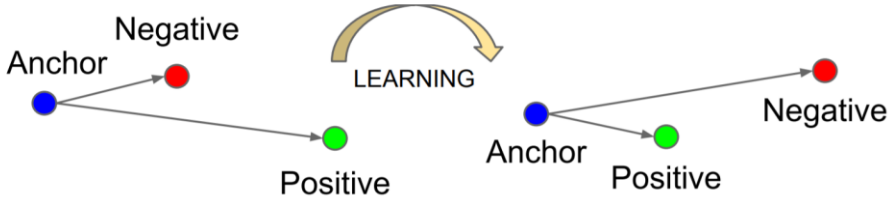


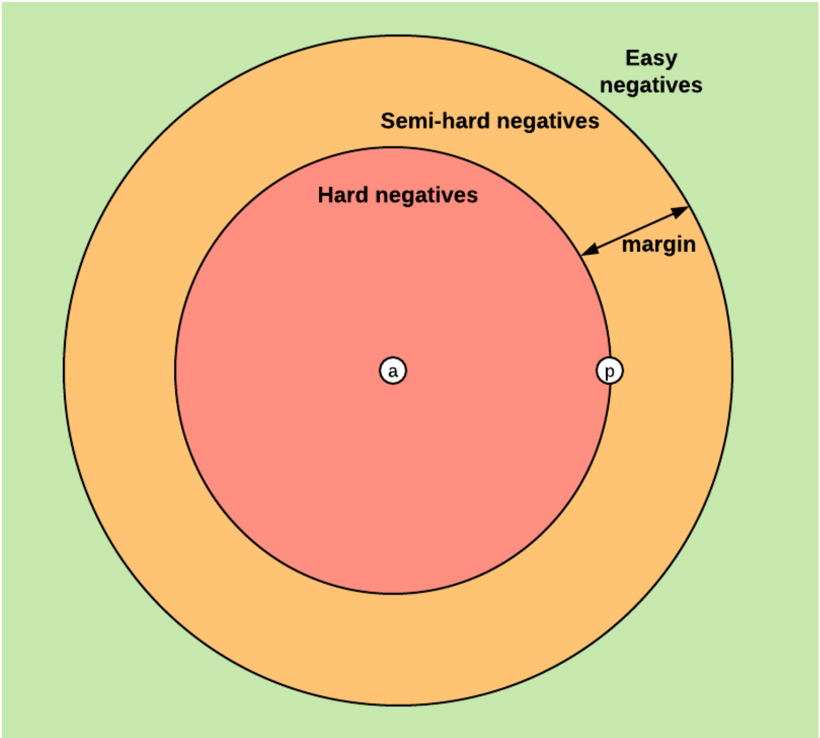
https://gombru.github.io/2019/04/03/ranking_loss/

Triplet Loss

- Triplet loss
  - Requires (anchor, positive, negative)
  - Anchor & negative should be further away than anchor & positive
  - To work well: requries good triplets

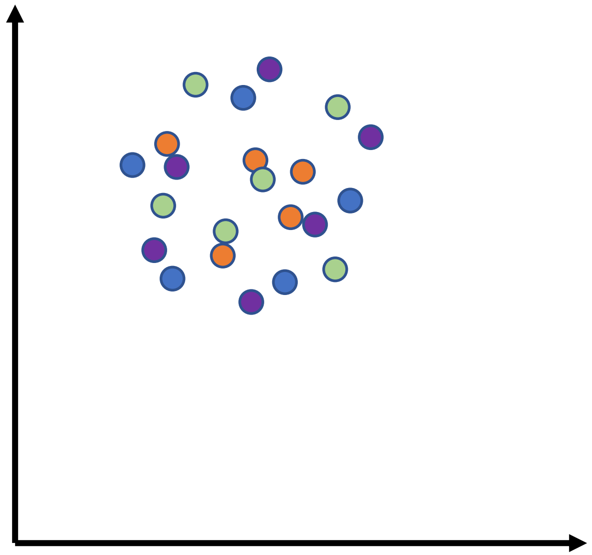




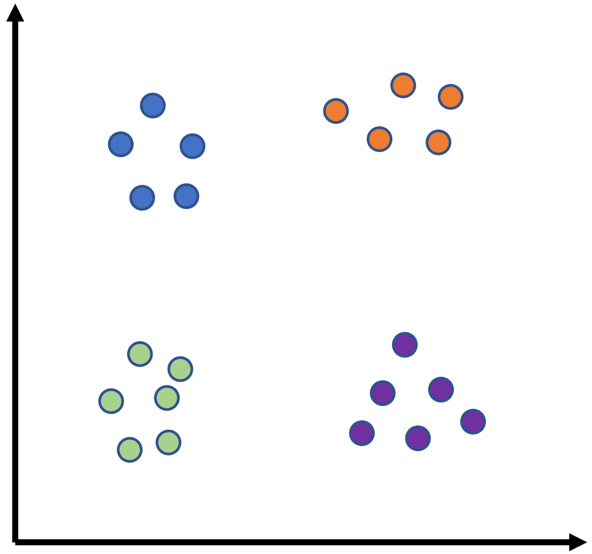


Global And Local Structure of Vector Space

- Global structure: Relation of two random sentences
- Local structure: Relation of two similar sentences
- Loss function must optimize local and global structure
- Constrative / Triplet loss might only optimize the local structure



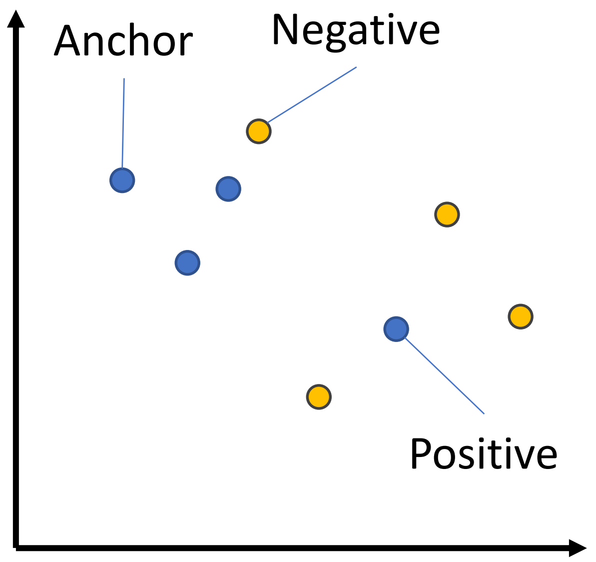
Bad global structure



Good global structure

Batch Hard Triplet Loss

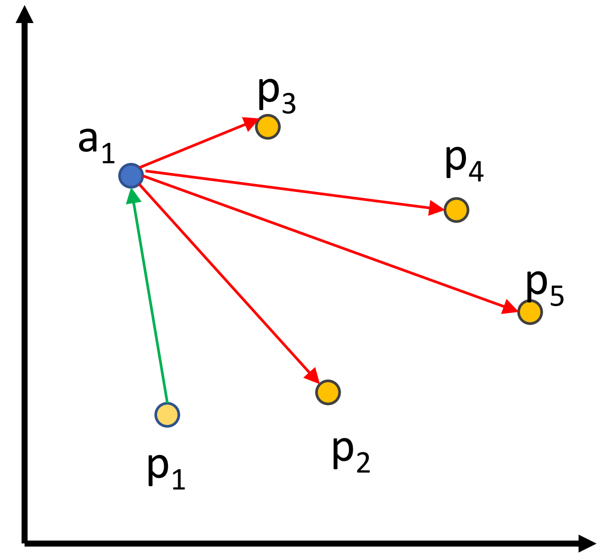
- Have a batch with multiple examples for each class
- Take an element as anchor
  - Take the farthest away element as positive
  - Take the closest element as negative
  - Train with triplet loss



- Loss automatically creates hard triplets

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

- Computed as ranking loss with Cross-Entropy:
  - Given a_1, which is the right answer out of $[p_1, p_2, p_3]$?
  - Compute scores: $[s(a_1, p_1), s(a_1, p_2), s(a_1, p_3)]$
  - Cross-Entropy loss with gold label: [1, 0, 0]

- Also called “training with in-batch negatives”, InfoNCE or NTXentLoss

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?

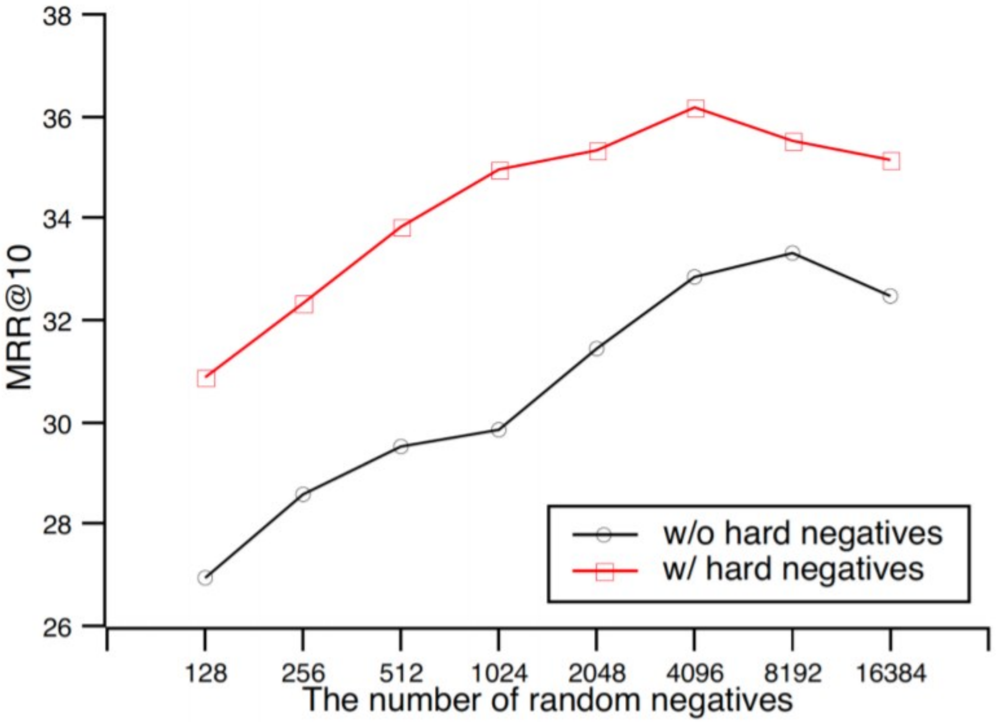
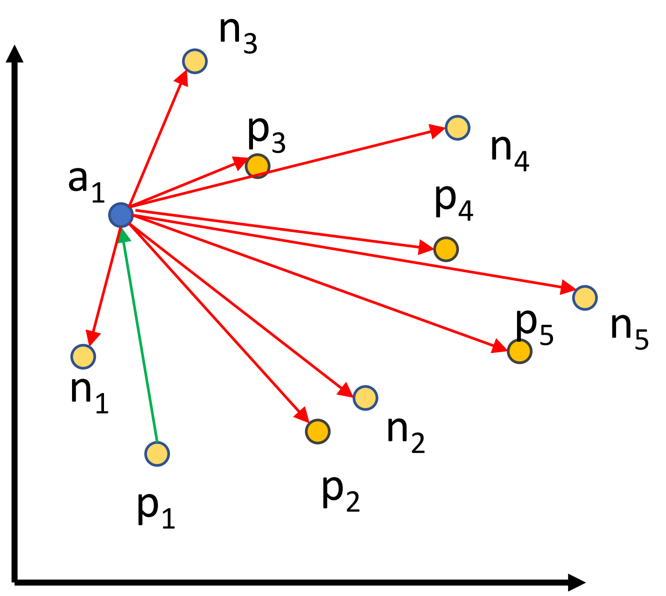


Image: https://arxiv.org/pdf/2010.08191.pdf

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.

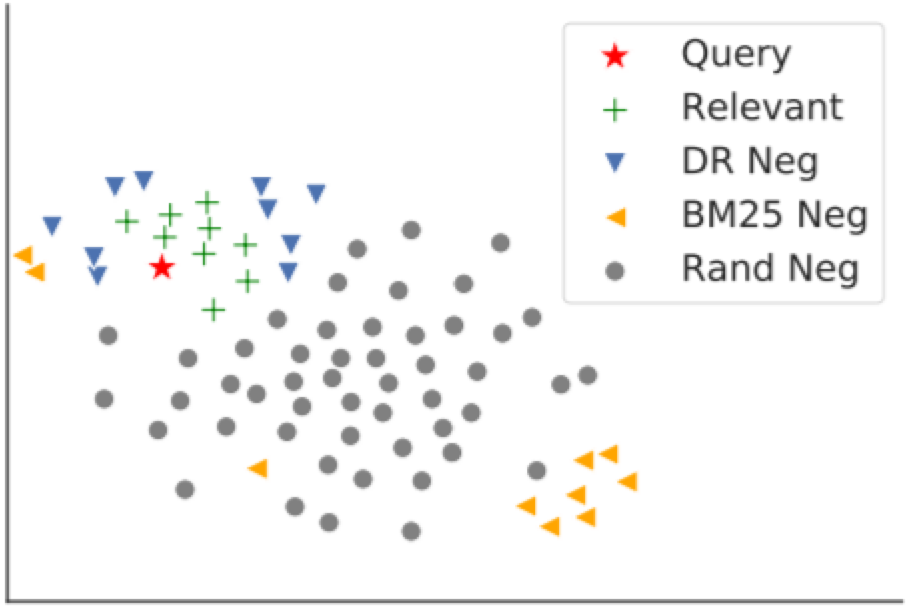
How to find hard-negatives?

- 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 most stars, Answer with less stars)

- Strategy 2: Mine hard negatives (simple):
  - Use BM25 to find top-100 most similar texts to anchor / positive
  - Select one of these randomly

Issue with BM25-Negatives

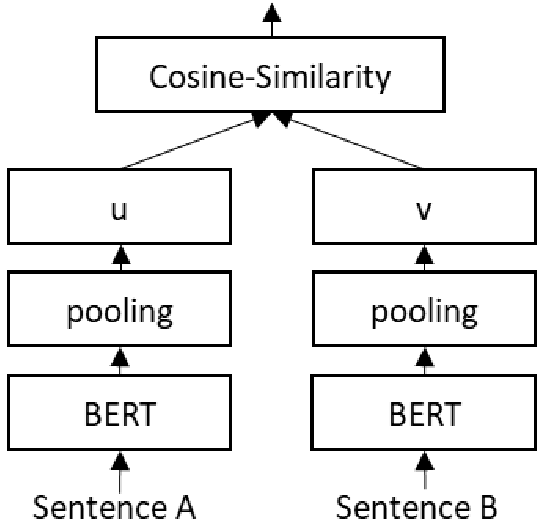


- BM25 negatives not necessarily hard negatives for the Dense Representations (DR)

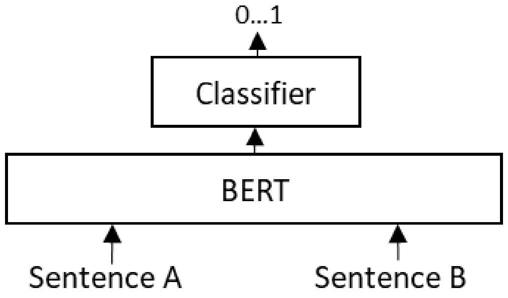
Img: https://arxiv.org/pdf/2007.00808.pdf

Bi- vs Cross-Encoders

Bi-Encoder



Cross-Encoder



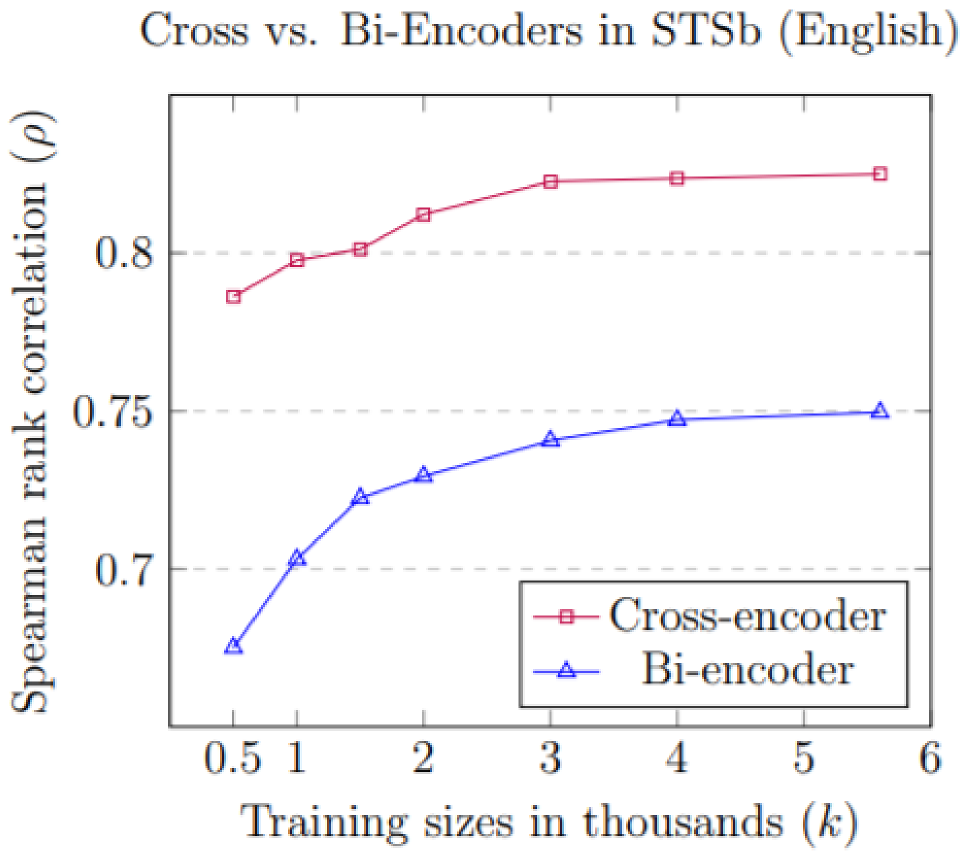
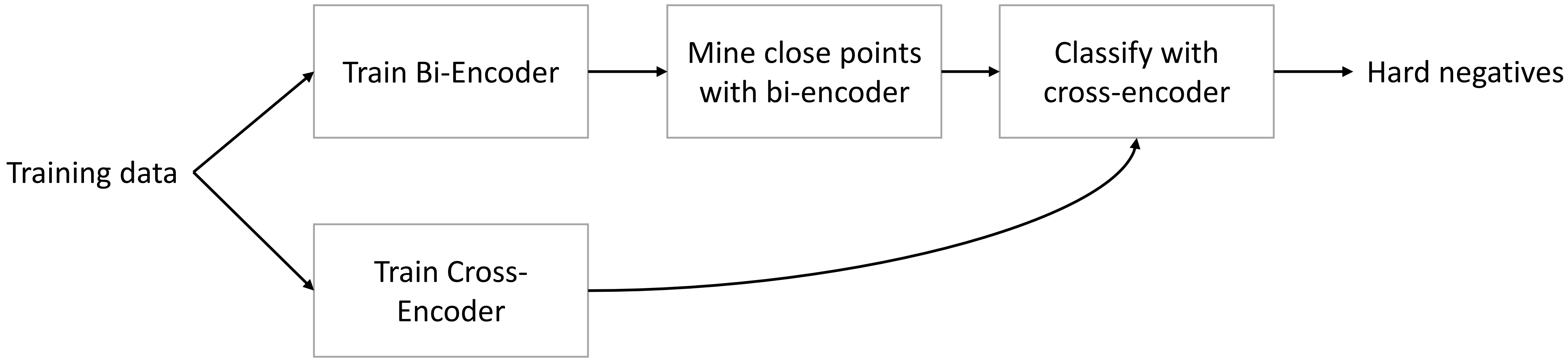


Image: https://arxiv.org/abs/2010.08240

How to find hard-negatives?



How big is the improvement?

Approach	**MRR@10 on MSMARCO Dev**
Random negatives	26.1
BM25 negatives	29.9
Mined hard negatives w/o denoising	26.0
Mined hard negatives with denoising	36.4

Source: https://arxiv.org/pdf/2007.00808.pdf and https://arxiv.org/pdf/2010.08191.pdf

Conclusion

- Many different loss functions available
  - In many cases, MultipleNegativeRankingLoss work well
- Adding hard negatives improves performance for search
  - but not for clustering!
- Finding hard negatives not trivial
  - Usage of powerful cross-encoders to mine those

Advanced Training Dense Text Representations

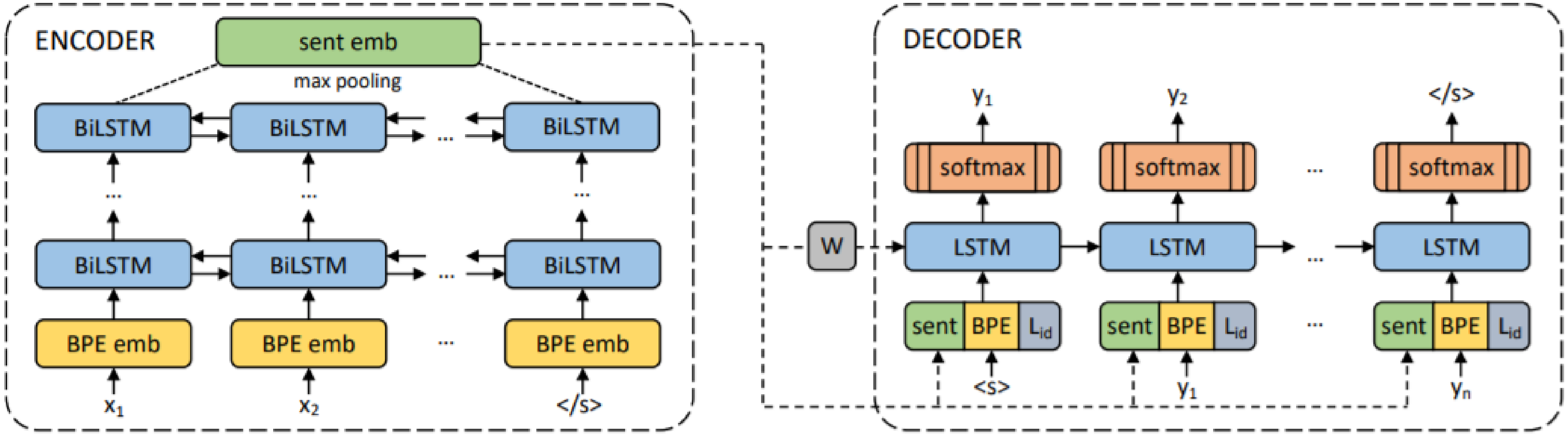
Nils Reimers

www.SBERT.net

Content

- Part 1: Introduction
- Part 2: Basic Training Methods
- **Part 3: Advanced Training Methods**
  - Multilingual Text Embeddings
  - Data Augmentation using Cross-Encoders
  - Unsupervised Embedding Learning
  - Neural Search / BEIR

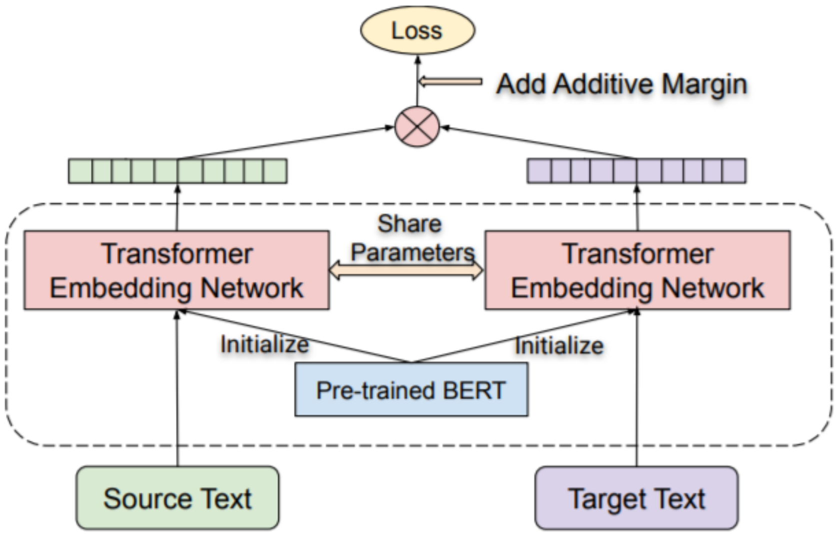
Multilingual Sentence Embeddings: LASER



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

https://arxiv.org/abs/1812.10464

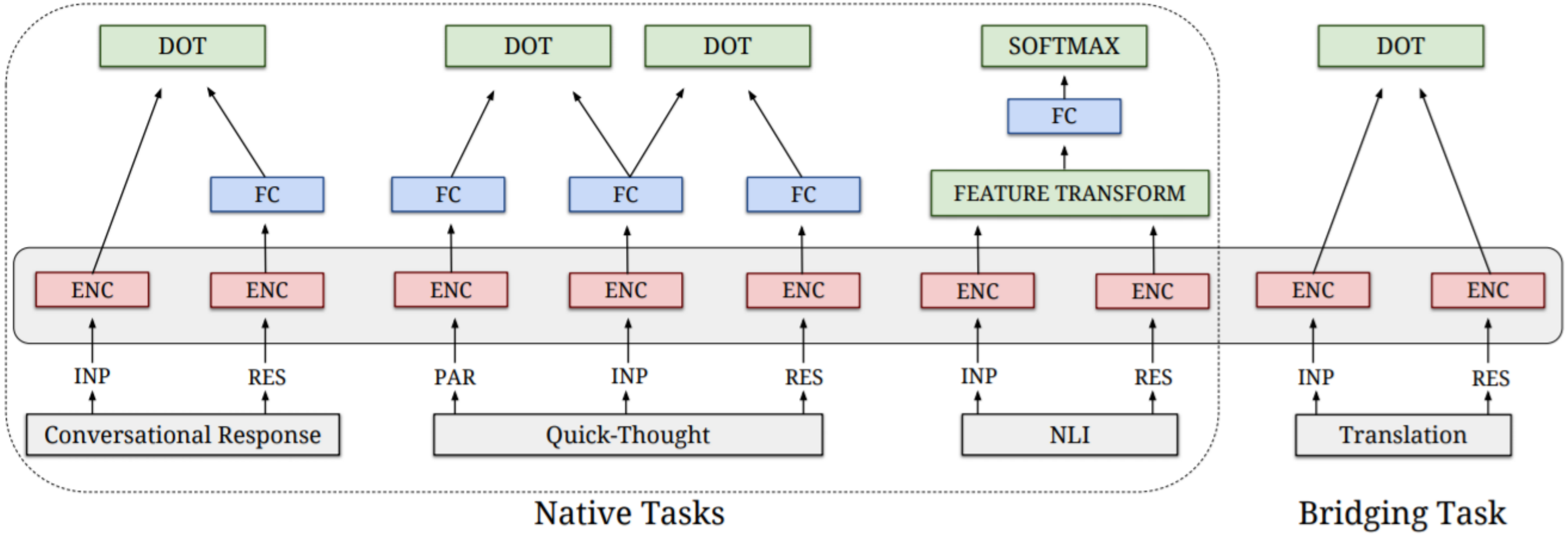
Multilingual Sentence Embeddings: LaBSE



- Translation ranking task
- Issues:
  - Cannot control what type of embeddings are learned
  - Works poorly on identifying similar sentences

https://arxiv.org/abs/2007.01852

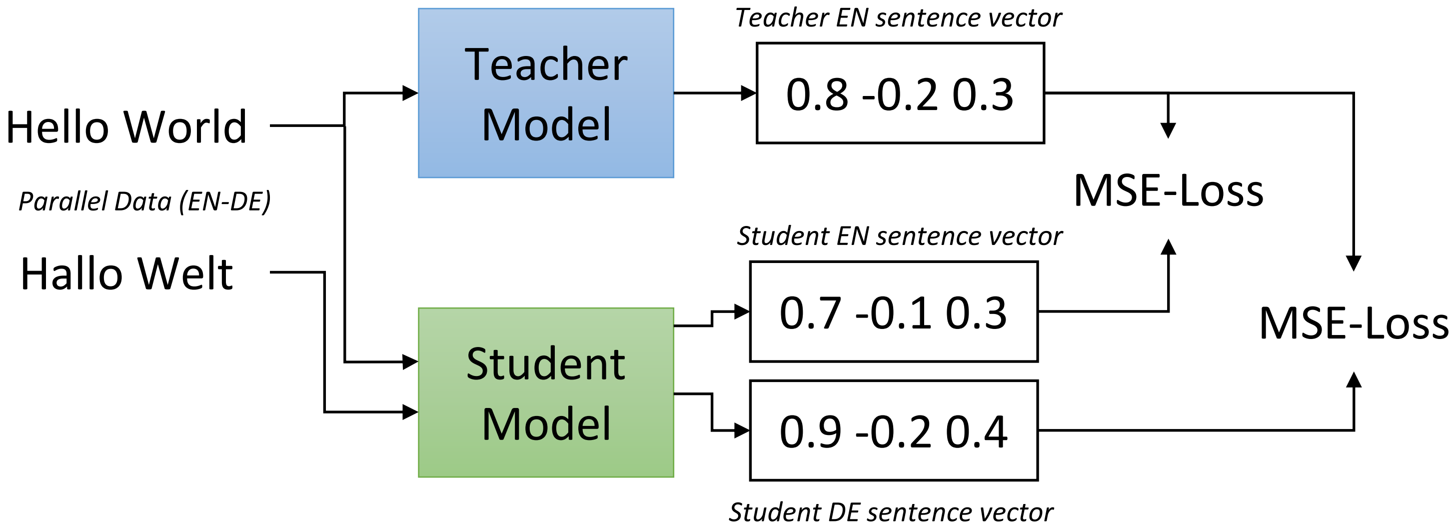
Multilingual Sentence Embeddings: mUSE



- 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

https://arxiv.org/abs/1810.12836

Multilingual Knowledge Distillation



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

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

https://arxiv.org/abs/2004.09813

Results – Multilingual Semantic Similarity

Model	EN-AR	EN-DE	EN-TR	EN-ES	EN-FR	EN-IT	EN-NL	Avg.
mBERT mean	16.7	33.9	16.0	21.5	33.0	34.0	35.6	27.2
XLM-R mean	17.4	21.3	9.2	10.9	16.6	22.9	26.0	17.8
mBERT-nli-stsb	30.9	62.2	23.9	45.4	57.8	54.3	54.1	46.9
XLM-R-nli-stsb	44.0	59.5	42.4	54.7	63.4	59.4	66.0	55.6
Knowledge Distillation								
mBERT ← SBERT-nli-stsb	77.2	78.9	73.2	79.2	78.8	78.9	77.3	77.6
DistilmBERT ← SBERT-nli-stsb	76.1	77.7	71.8	77.6	77.4	76.5	74.7	76.0
XLM-R ← SBERT-nli-stsb	77.8	78.9	74.0	79.7	78.5	78.9	77.7	77.9
XLM-R ← SBERT-paraphrases	82.3	84.0	80.9	83.1	84.9	86.3	84.5	**83.7**
Other Systems								
LASER	66.5	64.2	72.0	57.9	69.1	70.8	68.5	67.0
mUSE	79.3	82.1	75.5	79.6	82.6	84.5	84.1	81.1
LaBSE	74.5	73.8	72.0	65.5	77.0	76.9	75.1	73.5

- Training on English-only insufficient
- LASER & LaBSE perform badly

Bitext Mining

- Given two corpora: Find parallel (translated) sentences

Model	DE-EN	FR-EN	RU-EN	ZH-EN	Avg.
mBERT mean	44.1	47.2	38.0	37.4	41.7
XLM-R mean	5.2	6.6	22.1	12.4	11.6
mBERT-nli-stsb	38.9	39.5	26.4	30.2	33.7
XLM-R-nli-stsb	44.0	51.0	51.5	44.0	47.6
Knowledge Distillation					
XLM-R ← SBERT-nli-stsb	86.8	84.4	86.3	85.1	85.7
XLM-R ← SBERT-paraphrase	90.8	87.1	88.6	87.8	88.6
Other systems					
mUSE	88.5	86.3	89.1	86.9	87.7
LASER	95.4	92.4	92.3	91.7	93.0
LaBSE	95.9	92.5	92.4	93.0	93.5

Table 3: F_1 score on the BUCC bitext mining task.

- LASER & LaBSE better than mUSE & Knowledge Distillation
- Issue with mUSE & KD: They find similar sentences, that are not perfect translations

Data Efficiency

Dataset	#DE	EN-DE	#AR	EN-AR
XLM-R mean	-	21.3	-	17.4
XLM-R-nli-stsb	-	59.5	-	44.0
MUSE Dict	101k	75.8	27k	68.8
Wikititles Dict	545k	71.4	748k	67.9
MUSE + Wikititles	646k	76.0	775k	69.1

Dataset size	EN-DE	EN-AR
XLM-R mean	21.3	17.4
XLM-R-nli-stsb	59.5	44.0
1k	71.5	48.4
5k	74.5	59.6
10k	77.0	69.5
25k	80.0	70.2
Full TED2020	80.4	78.0

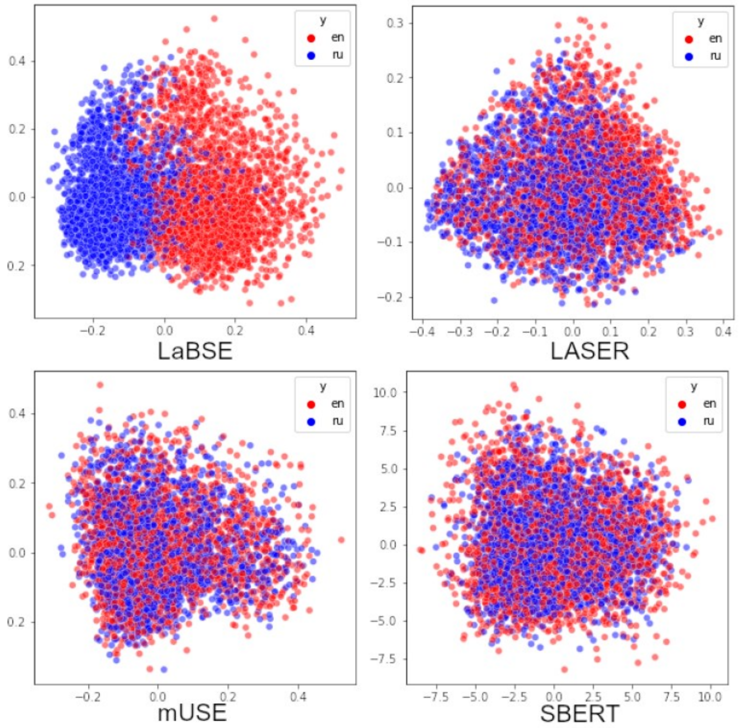
Table 6: Performance on STS 2017 dataset when trained with reduced TED2020 dataset sizes.

Knowledge Distillation vs. Training on Target Language

Model	KO-KO
LASER	68.44
mUSE	76.32
Trained on KorNLI & KorSTS	
Korean RoBERTa-base	80.29
Korean RoBERTa-large	80.49
XLM-R	79.19
XLM-R-large	81.84
Multiling. Knowledge Distillation	
XLM-R ← SBERT-nli-stsb	81.47
XLM-R-large ← SBERT-large-nli-stsb	83.00

Table 7: Spearman rank correlation on Korean STS-benchmark test-set (Ham et al., 2020).

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

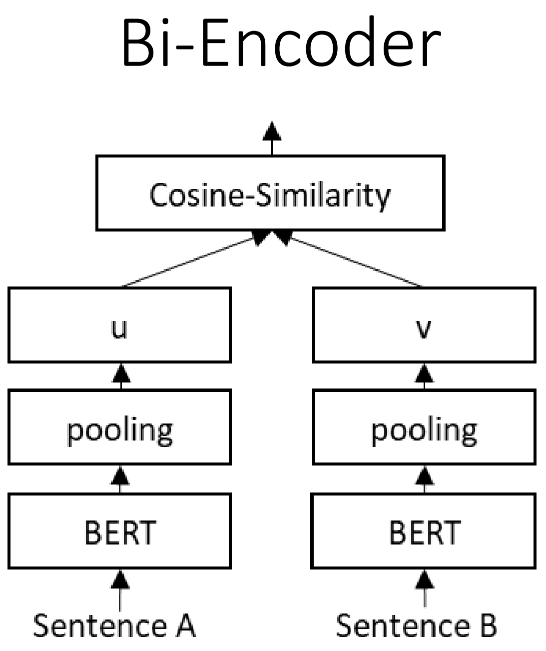
- Preference of certain language combinations
- Language bias impacts performance negatively on multilingual pools
- LASER and LaBSE with strong language bias

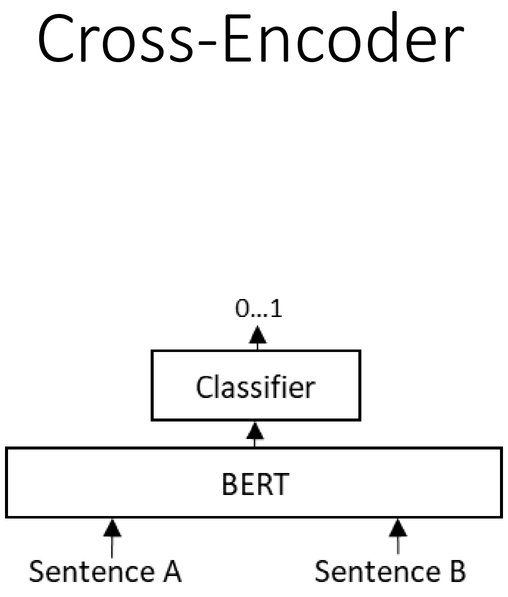
Augmented SBERT

Nils Reimers

www.SBERT.net

Bi- vs Cross-Encoders – Data Efficiency





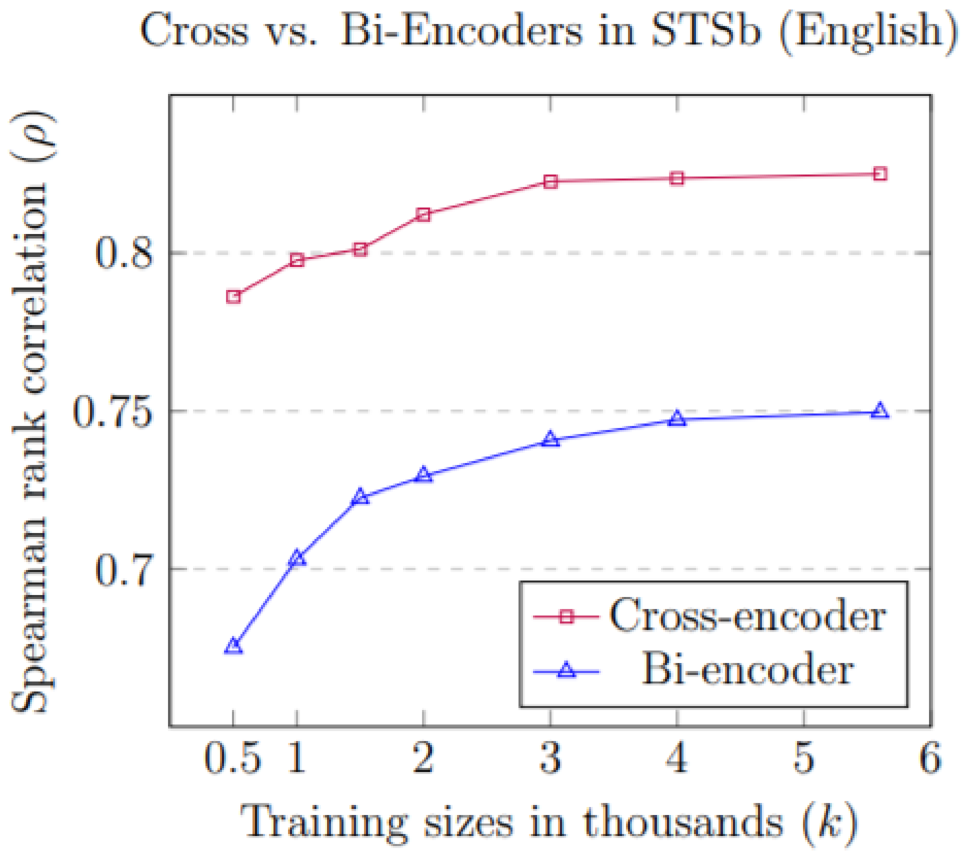
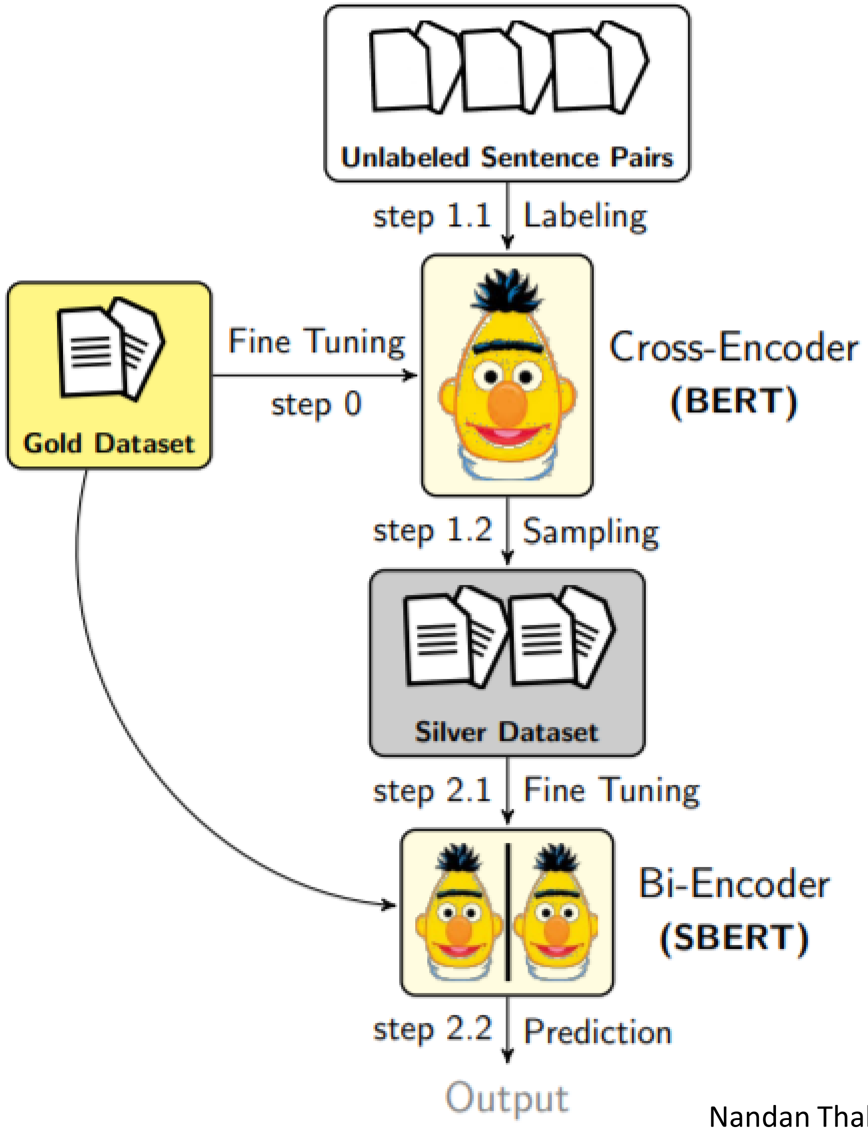


Image: https://arxiv.org/abs/2010.08240

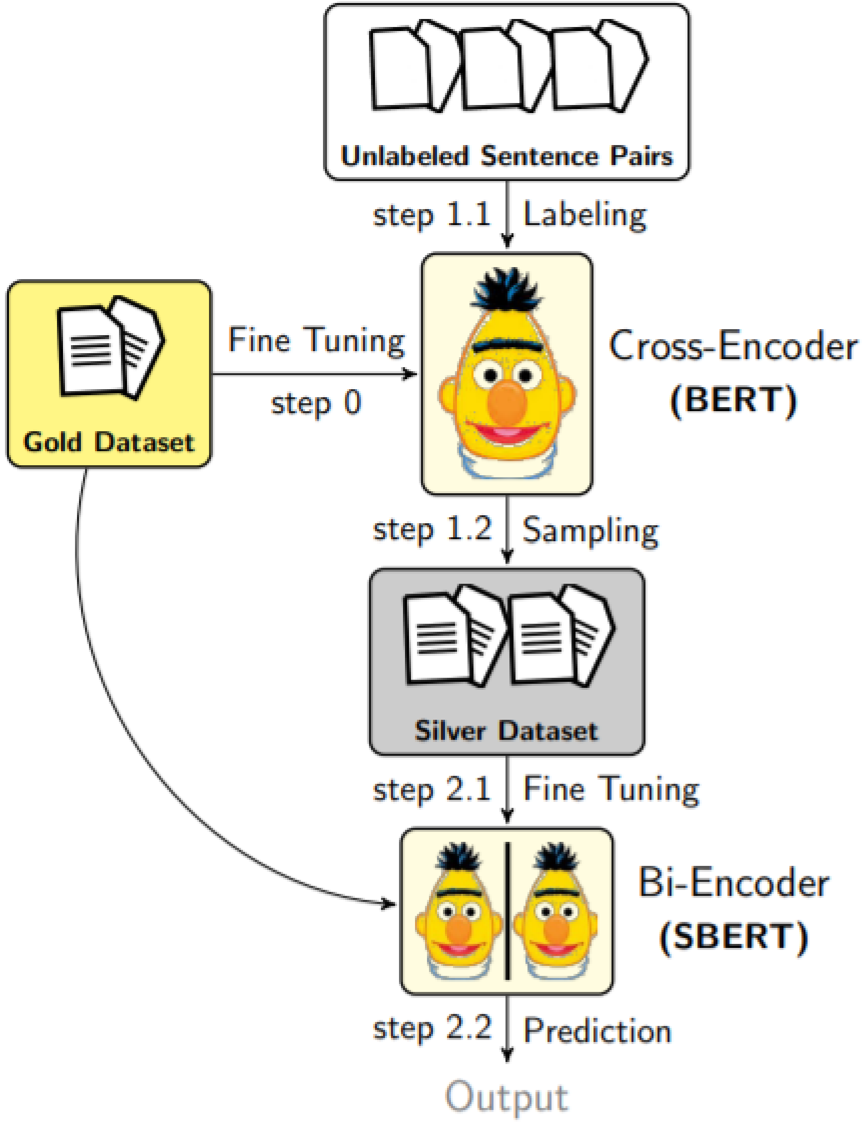
Augmented SBERT



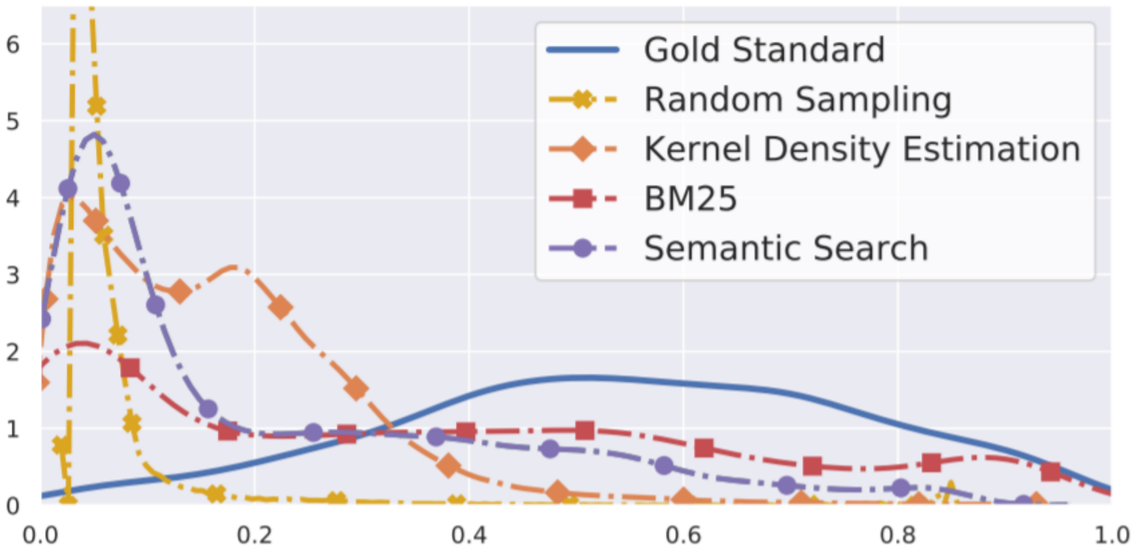
- Train Cross-Encoder
- Sample new sentence pairs
  - Label pairs with Cross-Encoder
- Train Bi-Encoder

Nandan Thakur, Nils Reimers, Johannes Daxenberger, Iryna Gurevych. Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks

Augmented SBERT - Sampling



- Random sampling yields mainly pairs with low similarity:



- BM25 sampling & Semantic Search yields best results

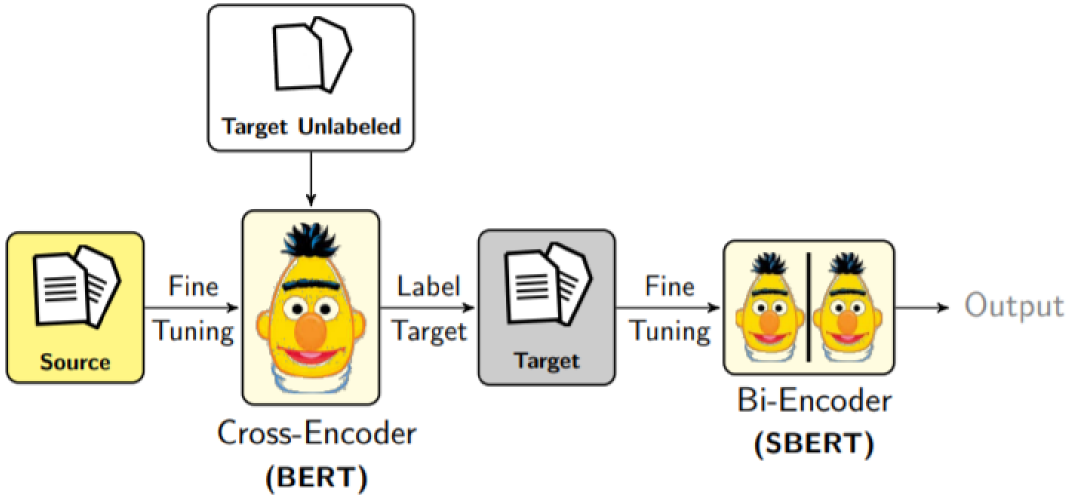
Augmented SBERT – In-Domain Results

Model	**Spanish-STS**	**Argument Similarity**	**Duplicate Questions**	**Paraphrase Identification**
BERT (Cross-Encoder)	77.5	65.1	80.4	89.0
SBERT (Bi-Encoder)	68.4	58.0	73.4	84.4
AugSBERT (Bi-Encoder)	**75.1**	**61.5**	**79.3**	**85.5**

In-domain improvement up to 6.7 points

Nandan Thakur, Nils Reimers, Johannes Daxenberger, Iryna Gurevych. Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks, NAACL 2021

Augmented SBERT – Cross-Domain



- Labeled training data on source domain only (Quora)
- Evaluation on specialized domains

Eval-Dataset	SBERT (Quora)	AugSBERT
AskUbuntu	50.1	60.2
Sprint	50.5	87.5
SuperUser	50.4	64.5

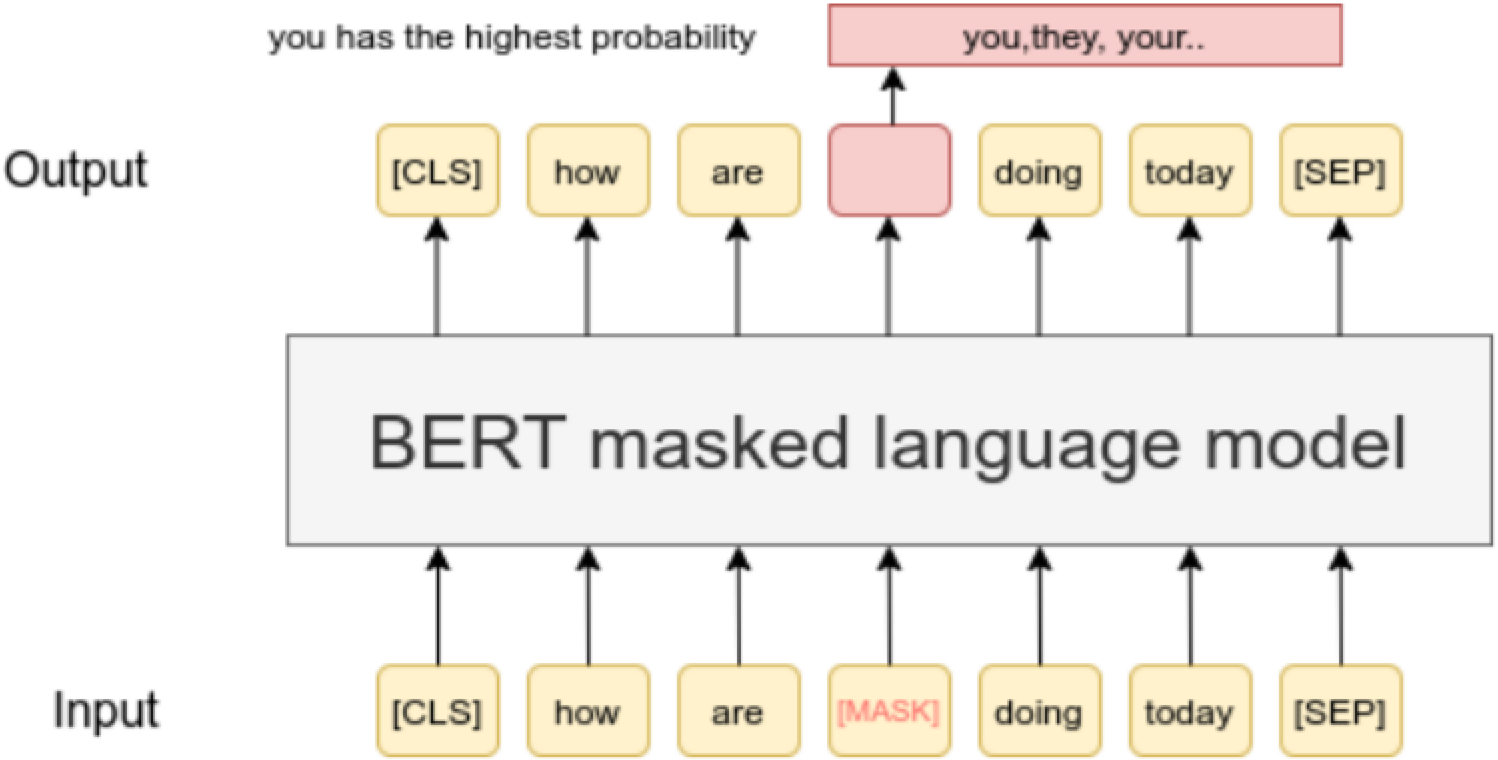
Cross-domain improvement up to 37 points

Unsupervised Embedding Methods

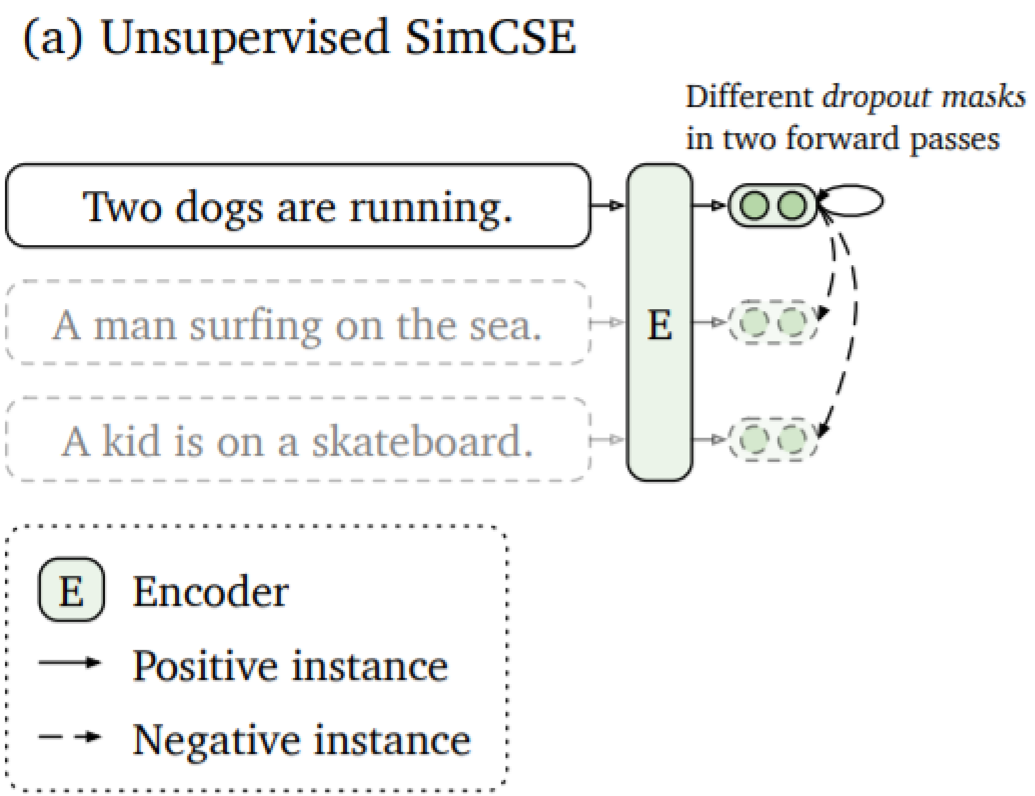
Nils Reimers

www.SBERT.net

Masked Language Model (MLM)



SimCSE

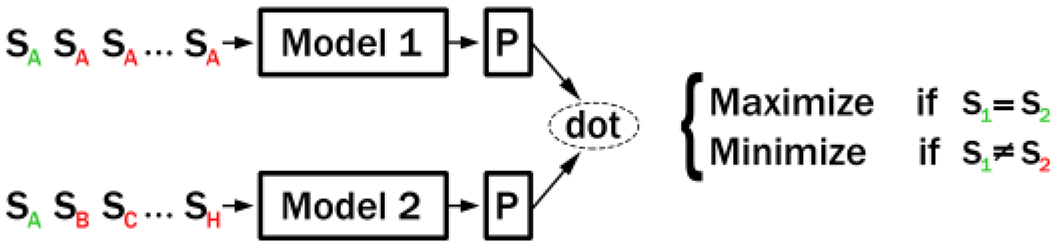


- Usage of MultipleNegativeRankingLoss
- Input pairs:
  (sent1, sent1)
  (sent2, sent2)
  ….

- Due to dropout: slightly different embeddings for f(sent1) and f(sent1)

https://arxiv.org/abs/2104.08821

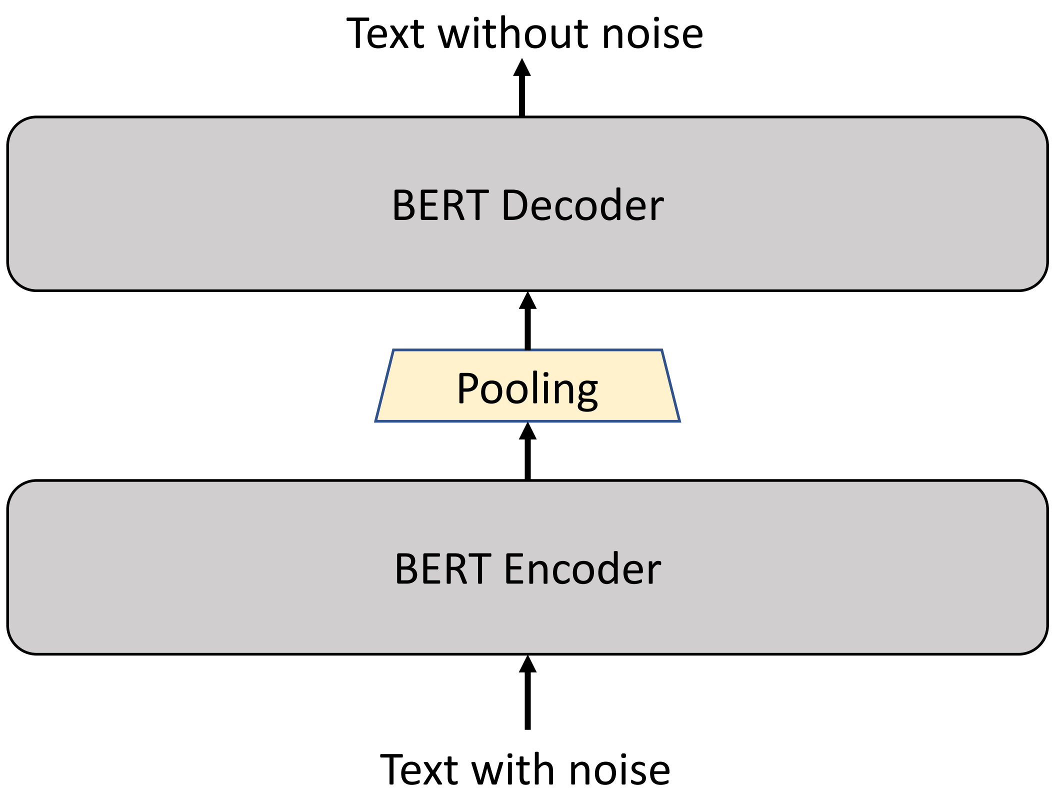
Contrastive Tension (CT)



- Initialize with two identical models
- Pass pairs with identical and with different sentences
- Maximize dot-score for identical sentences
- Minimize dot-score for different sentences

https://openreview.net/pdf?id=Ov_sMNau-PF

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

Issues in the Evaluation

- So far unsupervised methods evaluated on STS data

- Extremely bad way to evaluate unsupervised methods on STS datasets
  - Performance has near zero correlation to performance on real-world task
  - Simple sentences without domain specific knowledge
  - Unrealistic label distribution

- In TSDAE: Evaluation on domain specific datasets
  - AskUbuntu, StackExchange, Twitter, Scientific Publications

Evaluation

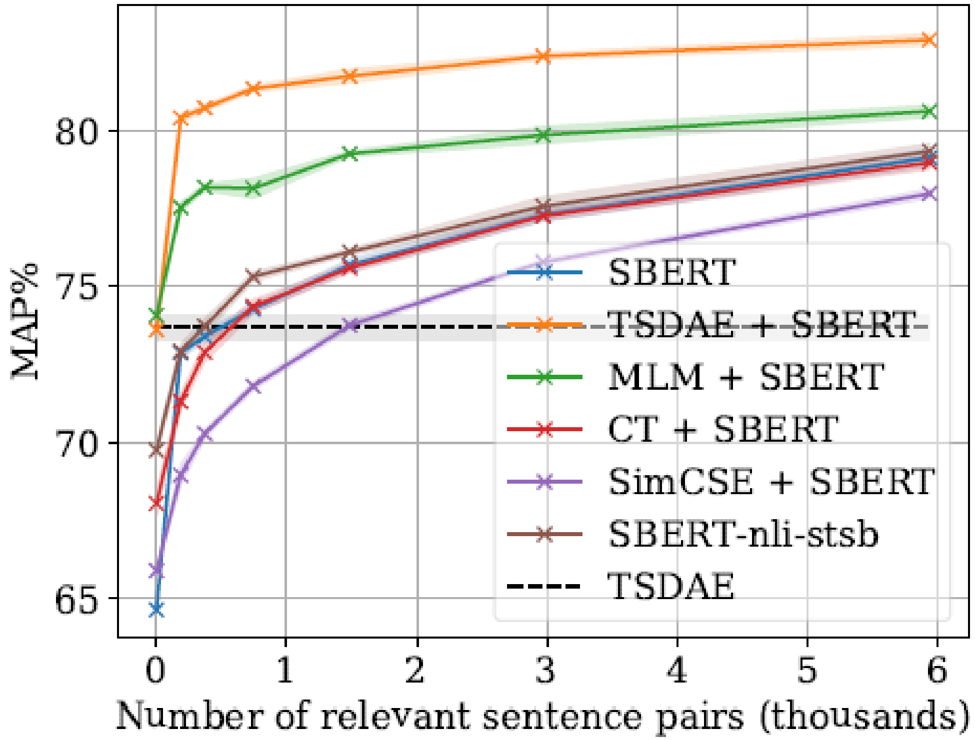
Method	**Avg. over 4 datasets**
TSDAE	55.2
MLM	52.9
CT	52.4
SimCSE	50.6
Out-of-the-box model	
SBERT on NLI+STSb	52.3

How good are unsupervised methods?

	AskUbuntu	**Twitter Paraphrases**	**StackExchange**	**SciDocs**
Unsupervised in-domain TSDAE on bert-base	55.6	74.1	36.2	**74.5**
Supervised out-of-domain mpnet + NLI + STSb	56.0	**78.9**	35.7	71.4
Supervised out-of-domain distilbert + MS MARCO	**56.1**	74.6	**40.3**	70.8

- Supervised pre-trained models hard to beat
- Diversity of pre-training dataset critical
  - Large, diverse dataset => great results across tasks

Unsupervised Method for Pre-Training



(d) SciDocs

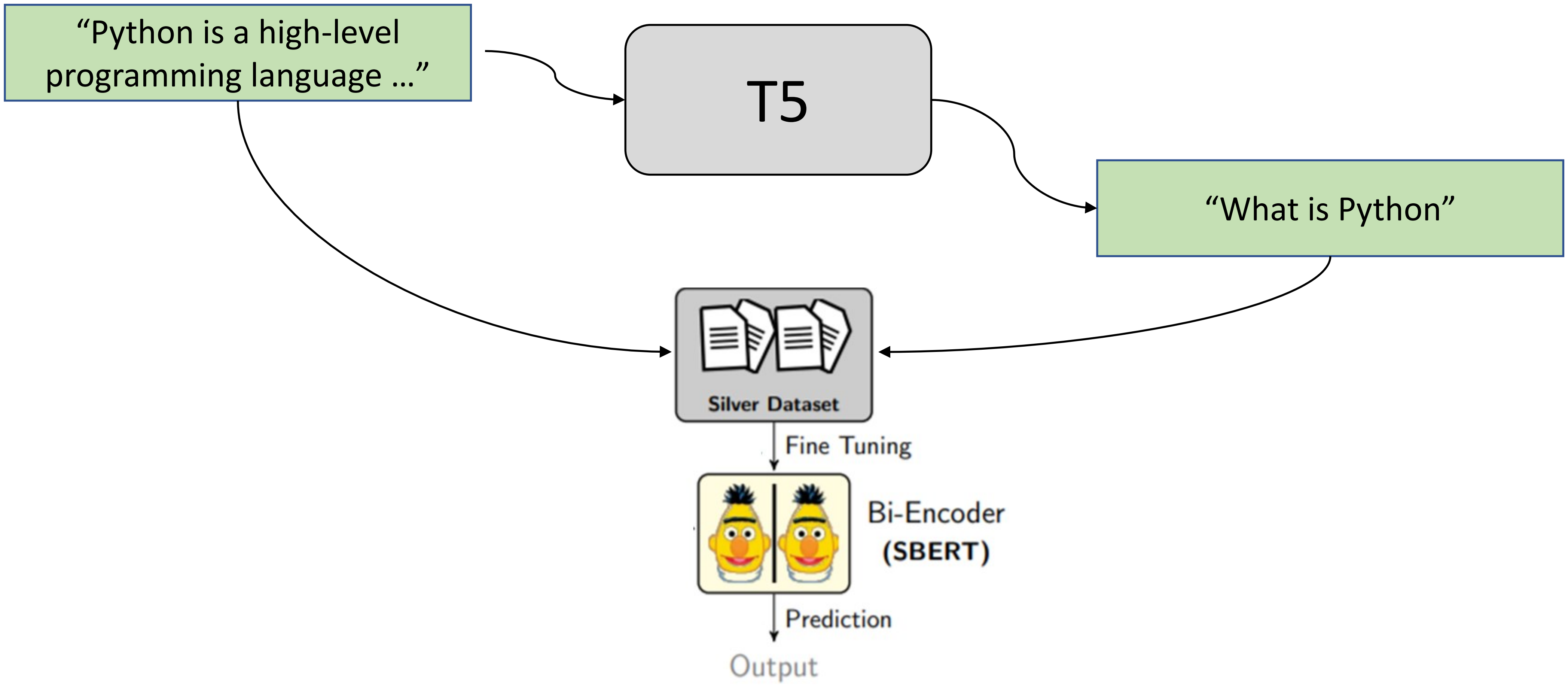
- Train unsupervised on large corpus from your domain
- Train supervised with some labels from your domain
- SimCSE / CT: Not helpful
- TSDAE / MLM: Big improvement

Domain Adaptation

Method	**Unsupervised**	**NLI+STS -> Unsupervised**	**Unsupervised -> NLI+STS**
TSDAE	55.2	54.2	**56.5**
MLM	52.9	51.1	**55.9**
CT	52.4	52.9	**53.0**
SimCSE	50.6	51.2	**52.4**

- First train unsupervised on your domain
- Then train supervised on available training data from other domains

GenQ: Synthetic Query Generation



https://arxiv.org/abs/2104.08663

Beir

Benchmarking IR

https://github.com/UKPLab/beir

Types of Search – BM25

How are you?

[0, 0, 0.3, 0, 0, 0, 0.1, 0, 0.2, …]

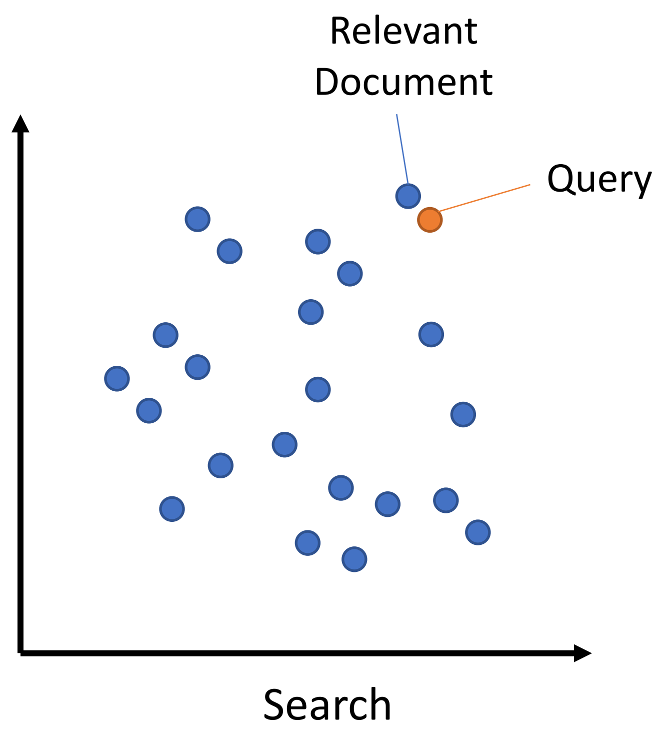
How are you

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

- Strong baseline
- Advantages:
  - Fast
  - Small index
  - No training required
- Disadvantages:
  - Lexical gap
  - Word order not preserved

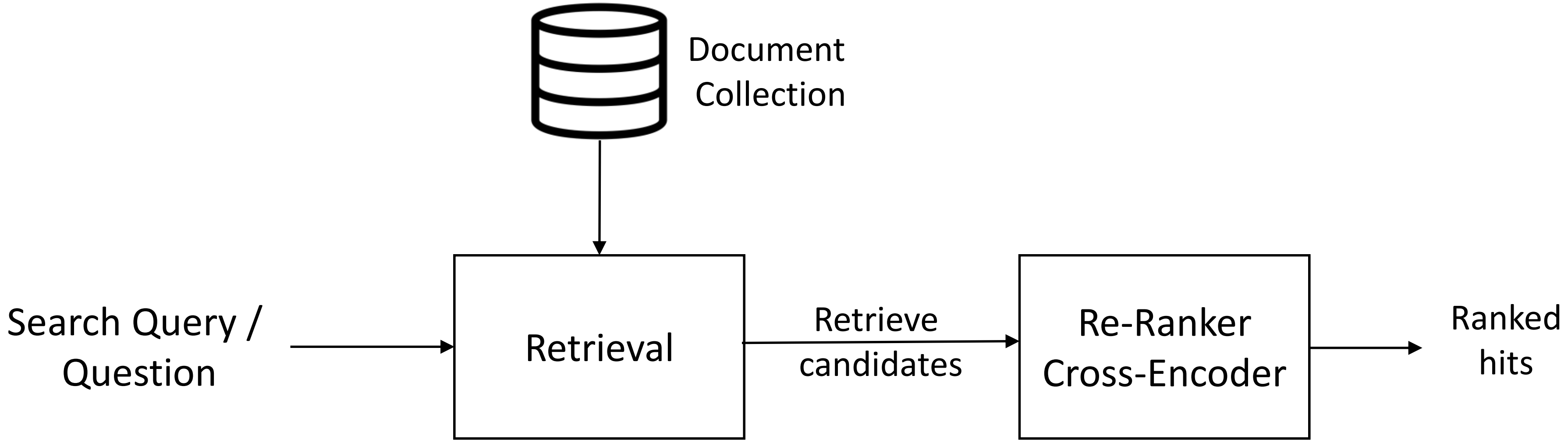
Img: https://en.wikipedia.org/wiki/Okapi_BM25

Types of Search – Dense Retrieval



- Maps query & docs to vector spaces
- Advantages:
  - Fast
  - Overcomes lexical gap
  - Can deal with word order
- Disadvantages:
  - Require training data

Types of Search- Re-Ranking

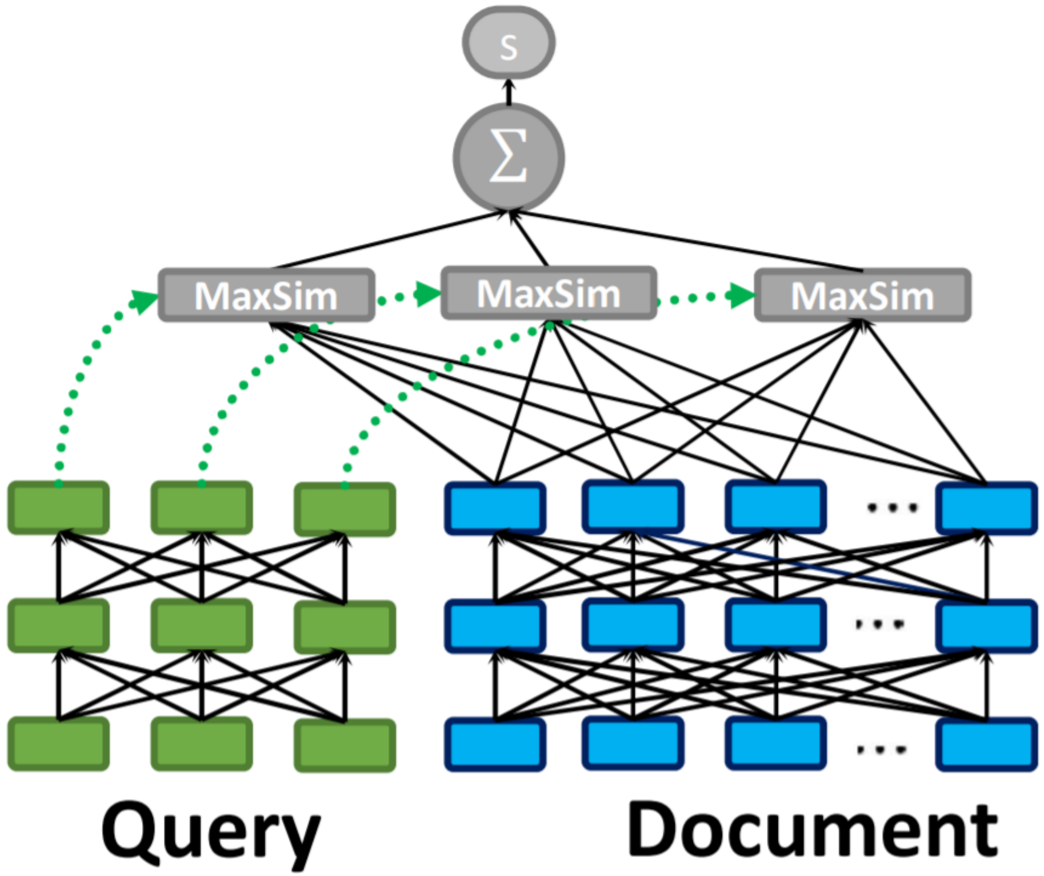


- Uses a Cross-Encoder to re-rank candidates

- Advantages:
  - High accuracy

- Disadvantages:
  - Slow
  - Requires training data

Types of Search- ColBERT



- Maps query & docs to token embeddings
- Advantages:
  - Good accuracy
- Disadvantages:
  - Large index size
  - Slow
  - Requires training data

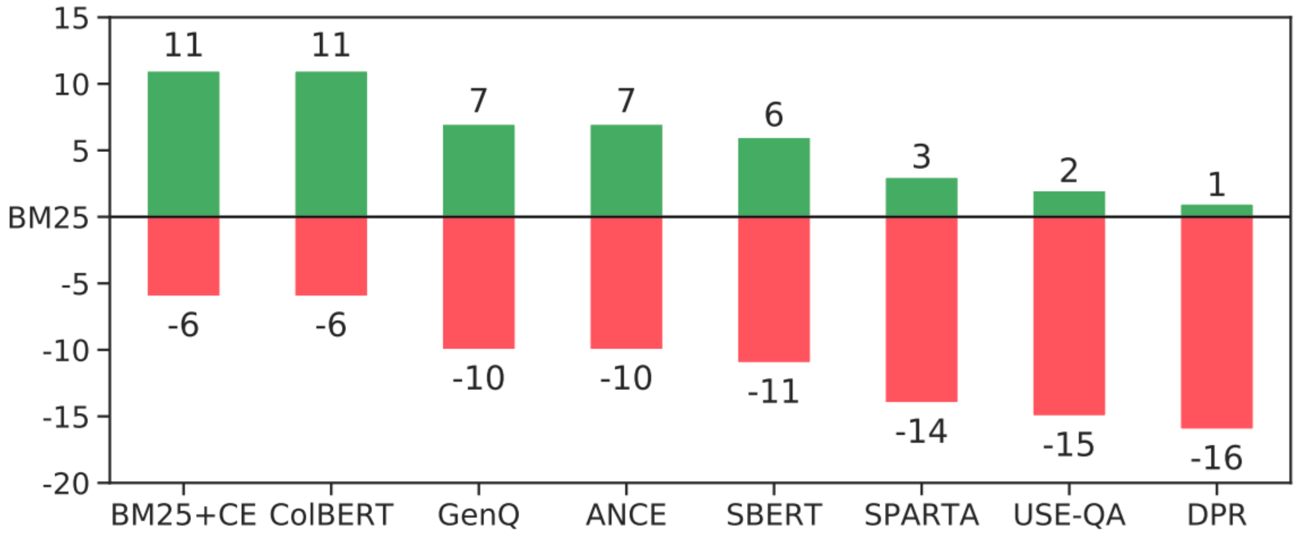
https://arxiv.org/pdf/2004.12832.pdf

Diverse Zero-Shot Evaluation of Neural Retrieval Methods

- How do neural retrieval models perform without in-domain training data?

https://arxiv.org/pdf/2104.08663.pdf

Which models generalize well?



BM25 (Lexical)

BM25 is an overall strong baseline for retrieval. It doesn't require to be fine-tuned for any dataset.

ColBERT, BM25 + CE (Rerank)

Reranking Models overall generalize well across diverse domains, and outperform BM25 on retrieval tasks

ANCE, GenQ, SBERT.. (Dense)

Dense models overall are unable to generalize. They perform well with a huge source overlap with target domain

https://arxiv.org/pdf/2104.08663.pdf

Speed – Quality – Memory-Trade-off

DBPedia (1 Million)			Retrieval Latency		Index
Rank	**Model**	**Dim.**	**GPU**	**CPU**	**Size**
(1)	BM25+CE	768	550ms	7100ms	0.4GB
(2)	ColBERT	128	350ms	–	20GB
(3)	BM25	–	–	20ms	0.4GB
(4)	GenQ	768	14ms	125ms	3GB
(5)	ANCE	768	20ms	275ms	3GB
(6)	SBERT	768	14ms	125ms	3GB
(7)	SPARTA	2000	–	20ms	12GB
(8)	USE-QA	512	35ms	75ms	2GB
(9)	DPR	768	19ms	230ms	3GB

- BM25+CE & ColBERT best systems
  - Slow / large index

- SBERT/ANCE better than BM25 if task similar to training data
  - Larger index (3GB vs. 0.4GB)
  - Slower on CPU (125ms vs 20ms)

https://arxiv.org/pdf/2104.08663.pdf

Conclusion

- Multilingual Knowledge Distillation
- Cross-Encoder useful to augment Bi-Encoder with additional data
- Unsupervised embeddings
  - Challenging
  - Performance not matching supervised methods
- Information retrieval
  - Trade-off: Memory-Speed-Quality