SBERT

SBERT (Sentence-BERT) in the context of LLMs refers to using Sentence-BERT as a semantic embedding model alongside or within Large Language Model (LLM) pipelines to improve understanding, retrieval, and reasoning over text.

SBERT vs LLM Embeddings

Feature	SBERT	LLM Embeddings
Speed	Very fast	Slower
Cost	Free (open-source)	Paid (API)
Size	Small	Large
Best for	Retrieval, similarity	Deep semantic understanding
Fine-tuning	Easy	Limited

Generating Contrastive Examples:

Generating contrastive examples in Large Language Models (LLMs) means creating pairs or sets of examples that are very similar in surface form but differ in meaning, label, or outcome, so that the model learns or is evaluated on fine-grained distinctions rather than shallow patterns.

A **contrastive example** consists of:

- **Positive (anchor / correct) example**
- **Negative (hard negative / contrastive) example**

They differ **minimally**, but their **labels or meanings change**.

Simple Example

"Bitcoin price increased due to high demand." → **Positive**

"Bitcoin price increased due to low demand." → **Negative**

Only one word changes, but the meaning flips.

Why Contrastive Examples Are Important in LLMs

Contrastive examples help LLMs:
Learn **robust semantic representations**
Reduce **spurious correlations**
Improve **reasoning and generalization**
Enhance **factual consistency**
Handle **edge cases and ambiguity**

Contrastive is especially useful in:

- Fine-tuning
- Prompt engineering
- Evaluation
- Alignment and safety

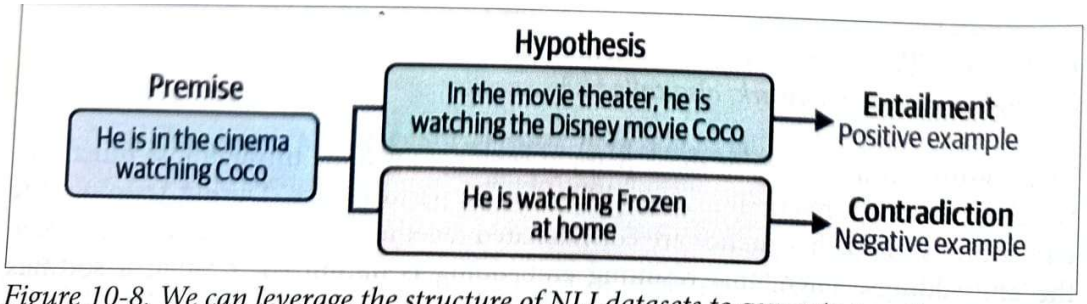

Figure 10-8. We can leverage the structure of NLI datasets to generate negative examples (contradiction) and positive examples (entailments) for contrastive learning.

Cosine Similarity:

Cosine similarity is a core mathematical measure used in Large Language Models (LLMs) to compare text embedding's and determine how semantically similar two pieces of text are.

Why Cosine Similarity Is Used in LLMs

LLMs convert text into **high-dimensional embedding's**.
Cosine similarity is preferred because:

Length-independent
Works well in high-dimensional spaces
Stable for semantic comparison
Computationally efficient

Cosine Similarity Vs Other Distance Metrics:

Metric	Measures	Used When
Cosine Similarity	Angle	Semantic meaning
Euclidean Distance	Absolute distance	Magnitude matters
Dot Product	Projection	Fast but unnormalized
Manhattan Distance	L1 distance	Sparse vectors

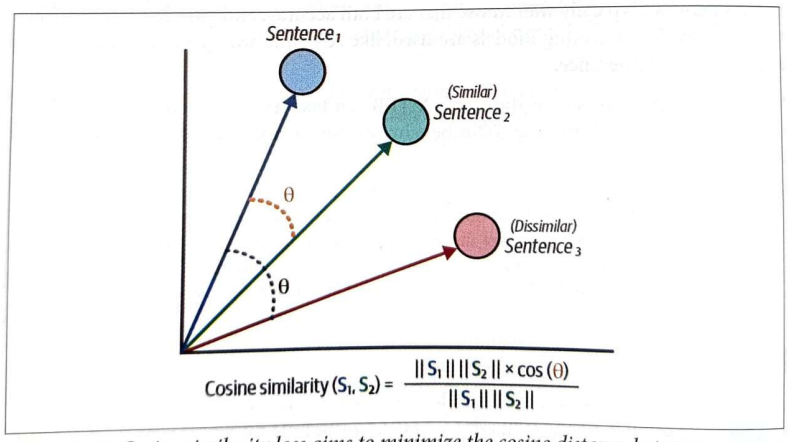

Figure 10-9. Cosine similarity loss aims to minimize the cosine distance between semantically similar sentences and to maximize the distance between semantically dissimilar sentences.

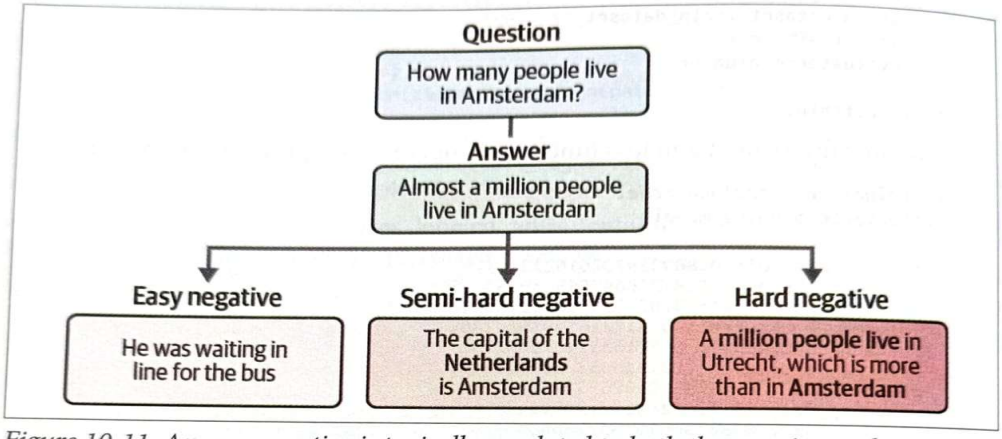

Figure 10-11. An easy negative is typically unrelated to both the question and answer. A semi-hard negative has some similarities to the topic of the question and answer but is somewhat unrelated. A hard negative is very similar to the question but is generally the wrong answer.

Gathering negatives can roughly be divided into the following three processes:

Easy negatives
: Through randomly sampling documents as we did before.

Semi-hard negatives
: Using a pretrained embedding model, we can apply cosine similarity on all sentence embeddings to find those that are highly related. Generally, this does not lead to hard negatives since this method merely finds similar sentences, not question/answer pairs.

Hard negatives
: These often need to be either manually labeled (for instance, by generating semi-hard negatives) or you can use a generative model to either judge or generate sentence pairs.

Make sure to *restart* your notebook so we can explore the different methods of fine-tuning embedding models.

Augmented SBERT

Augmented SBERT refers to enhancing Sentence-BERT (SBERT) using LLMs, external knowledge, or advanced training strategies to produce richer, more robust sentence embedding's for downstream LLM applications such as RAG, semantic search, clustering, and topic modelling.

In short:
SBERT = base semantic encoder
Augmented SBERT = SBERT + LLM/knowledge/data augmentation

Why Do We Need Augmented SBERT?

Standard SBERT embedding's may struggle with:

- Domain-specific language (finance, medicine)
- Logical negation and reasoning
- Sparse or low-resource data
- Hard negative examples

Augmentation improves embedding quality and robustness.

Architecture: Augmented SBERT in LLM Systems:

```
Text / Query
     ↓
LLM Augmentation
(paraphrase, contrast, expand)
     ↓
Augmented Text
     ↓
SBERT Encoder
     ↓
Enhanced Embeddings
     ↓
Vector DB / RAG / BERTopic
```

Key Advantages:

Faster than LLM embedding's
Near-LLM-level semantic quality

Scalable and cost-effective
Ideal for RAG and BERTopic

Aspect	SBERT	Augmented SBERT
Training data	Generic	LLM-augmented
Domain knowledge	Limited	High
Negation handling	Moderate	Improved
Robustness	Medium	High
Retrieval quality	Good	Excellent

Here, a gold dataset is a small but fully annotated dataset that holds the ground truth. A silver dataset is also fully annotated but is not necessarily the ground truth as it was generated through predictions of the cross-encoder.

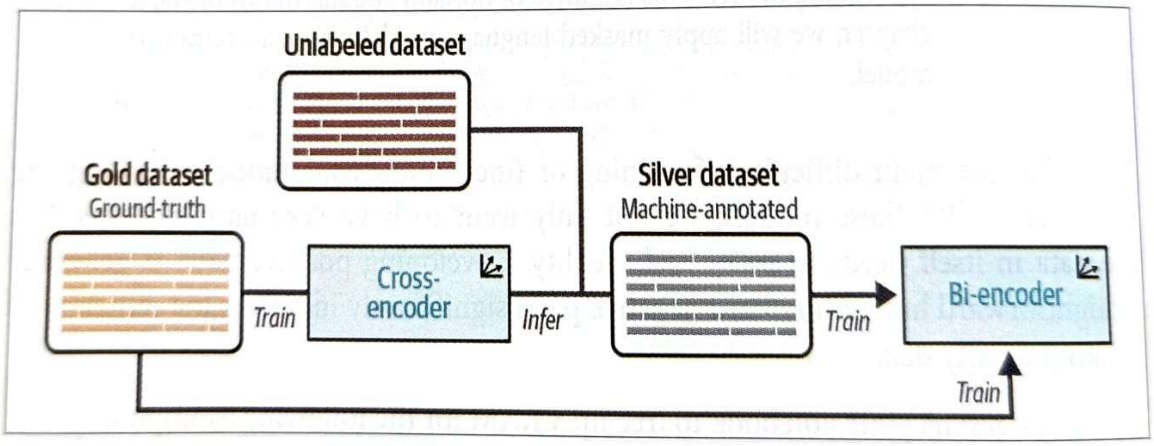

Figure 10-12. Augmented SBERT works through training a cross-encoder on a small gold dataset, then using that to label an unlabeled dataset to generate a larger silver dataset. Finally, both the gold and silver datasets are used to train the bi-encoder.

Supervised Learning

Supervised learning in LLMs uses labelled data, where the correct output (answer, label, or response) is explicitly provided for each input.

Where Supervised Learning Appears in LLMs

1. Supervised Fine-Tuning (SFT)

After pretraining, LLMs are fine-tuned on **instruction–response pairs**.

Example

Input: Explain cosine similarity in NLP.

Output: Cosine similarity measures the angle...

Improves instruction-following
Aligns model behaviour with human expectations

2. Task-Specific Fine-Tuning

Used for:

- Sentiment classification
- NLI
- Named Entity Recognition
- Question answering

3. Embedding Models (SBERT)

SBERT uses **supervised contrastive learning**:

- Positive pairs
- Negative pairs
- Labels indicate similarity or class

4. Human Feedback (RLHF – Partial Supervision)

- Human-labelled preferences
- Rankings or scores guide model optimization

Advantages

✓ High accuracy
✓ Clear objective
✓ Faster convergence

Unsupervised Learning:

Unsupervised learning in LLMs learns patterns from unlabelled data without explicit target outputs.

Where Unsupervised Learning Appears in LLMs

1. Pretraining (Core of LLMs)

LLMs are primarily pretrained using self-supervised learning, a form of unsupervised learning.

Objective

- Predict next token
- Masked language modeling

Example:

"The price of Bitcoin is [MASK]."

2. Representation Learning

- Learning grammar, syntax, semantics
- Capturing world knowledge

3. Topic Modeling & Clustering

- BERTopic
- Document clustering
- Semantic grouping

4. Unsupervised Embedding Training

- Sentence embedding's without labels
- Contrastive learning with in-batch negatives

Advantages

✓ Scales to massive data
✓ No labeling cost
✓ Learns general language structure

Self-Supervised Learning (Bridge Between Both)

LLMs mainly use self-supervised learning, which is:

- Unsupervised in data usage
- Supervised in training signal

Example

- Input = context tokens
- Label = next token (automatically generated)

Supervised vs Unsupervised: Comparison Table

Aspect	Supervised Learning	Unsupervised Learning
Labels	Required	Not required
Main stage	Fine-tuning	Pretraining
Data scale	Smaller	Massive
Cost	High	Low
Control	High	Low
Examples	SFT, SBERT	GPT pretraining