What is BERT ?

<https://huggingface.co/blog/bert-101?utm_source=chatgpt.com#6-the-open-source-power-of-bert>

<https://mccormickml.com/2019/11/11/bert-research-ep-1-key-concepts-and-sources/?utm_source=chatgpt.com>

<https://mccormickml.com/2019/07/22/BERT-fine-tuning/#31-bert-tokenizer>

BERT is pre-trained on a large corpus of unlabeled text

data using two unsupervised tasks: Masked Language

Modeling (MLM) and Next Sentence Prediction. This

pre-training allows BERT to develop deep bidirectional

representations of language.

Unlike traditional language models that process text

sequentially, BERT uses the transformer architecture to

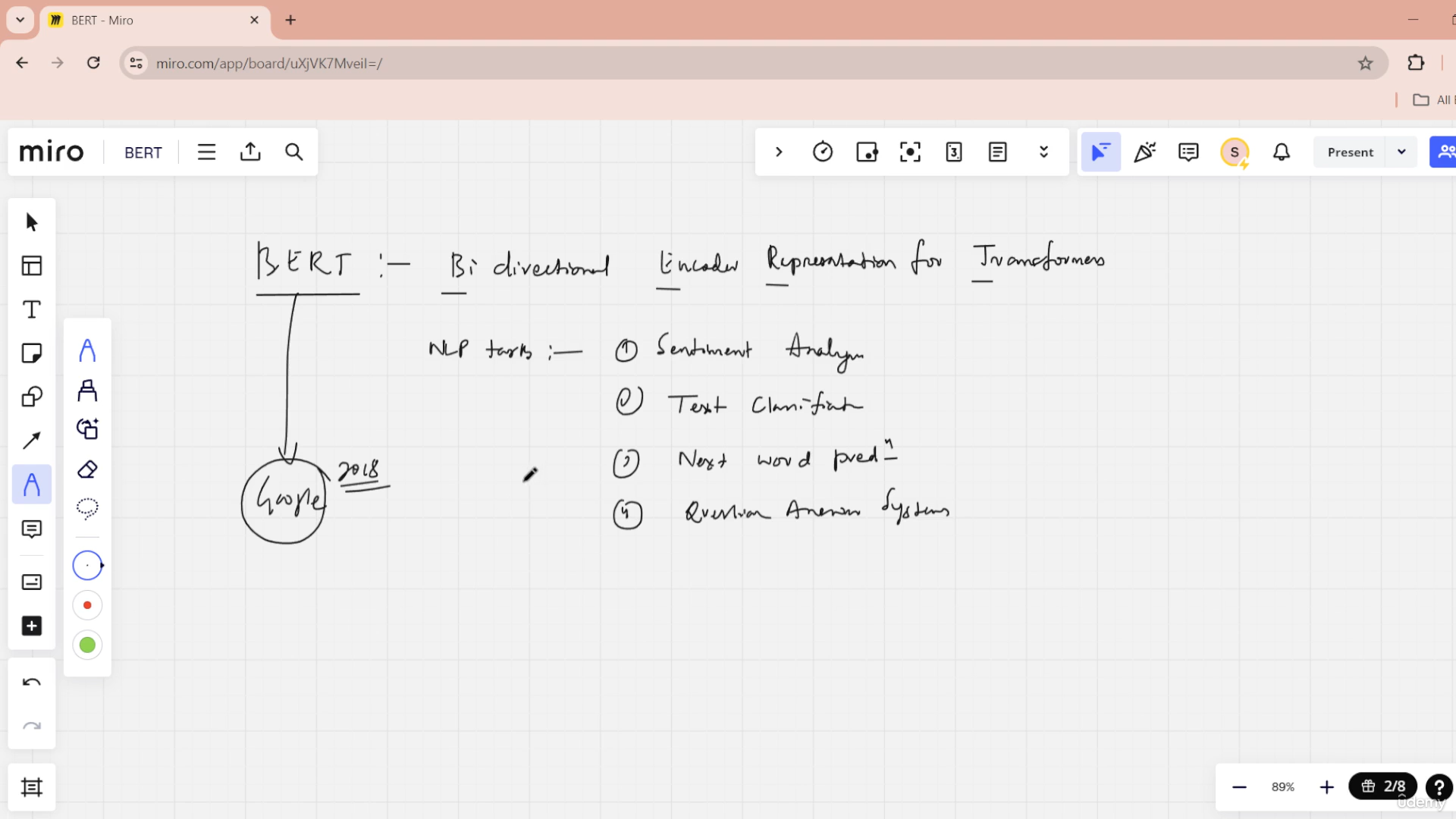
attend to all words in a sentence simultaneously,

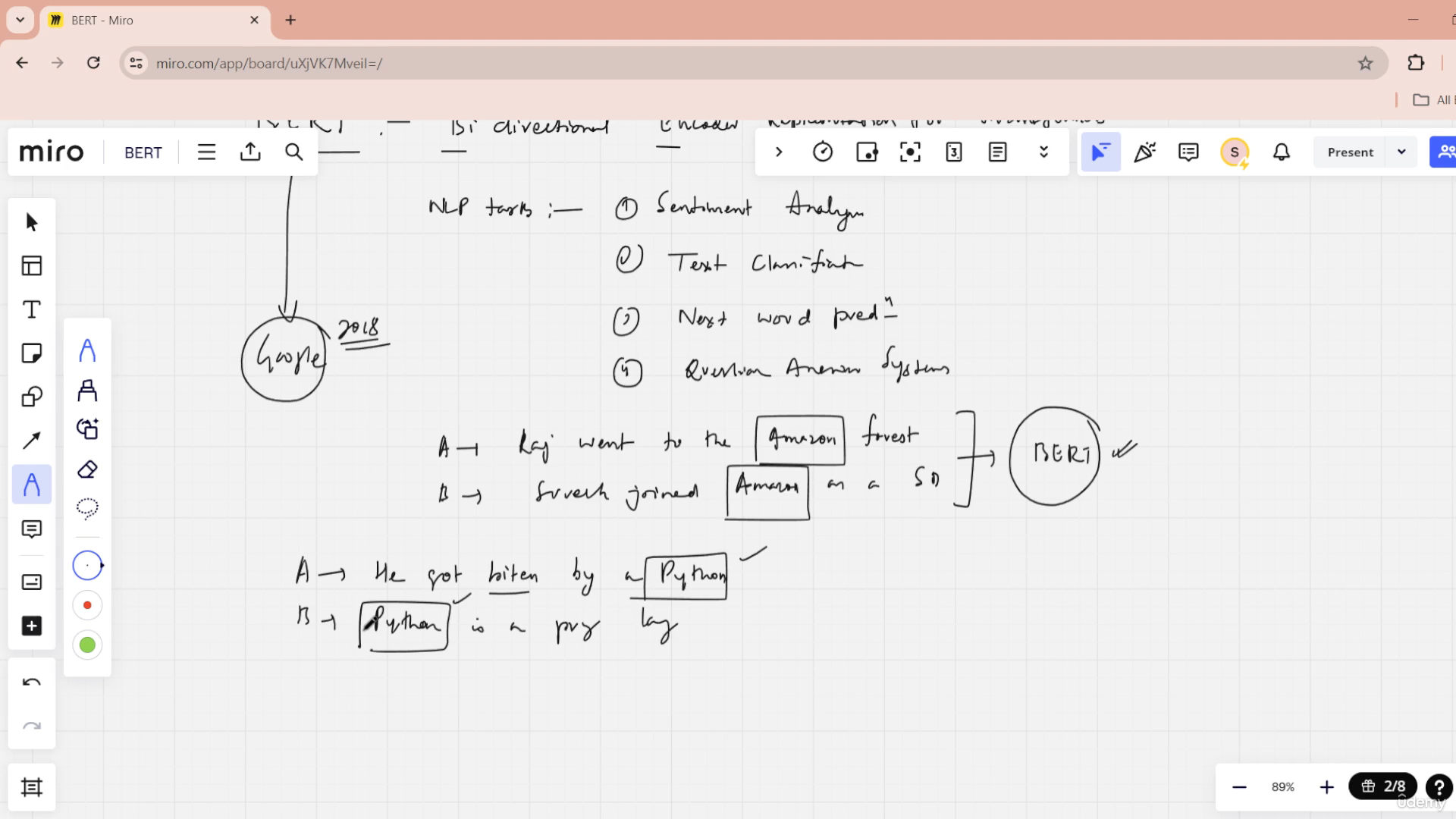
allowing it to capture bidirectional context.

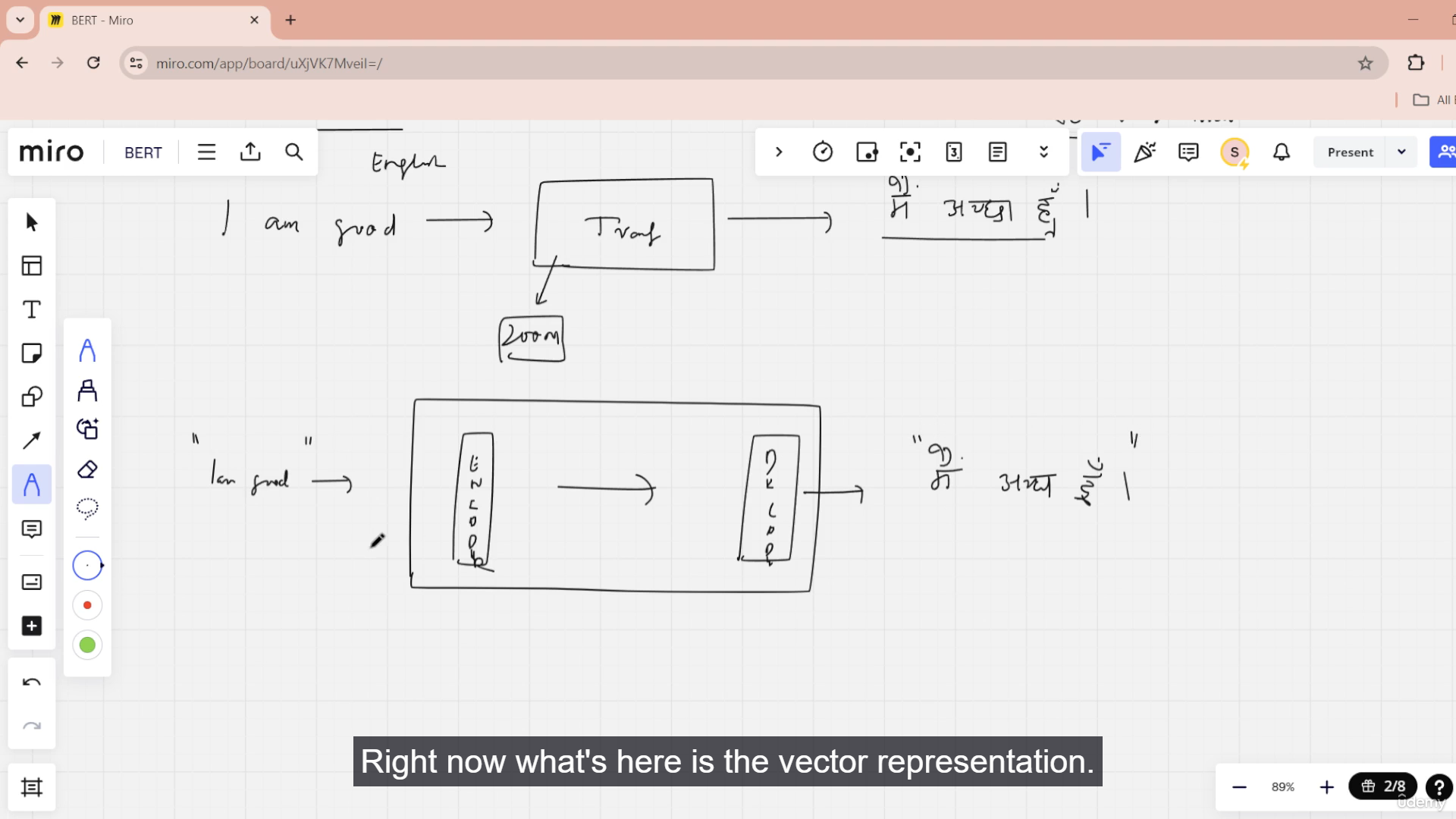
BERT's architecture is based on the transformer

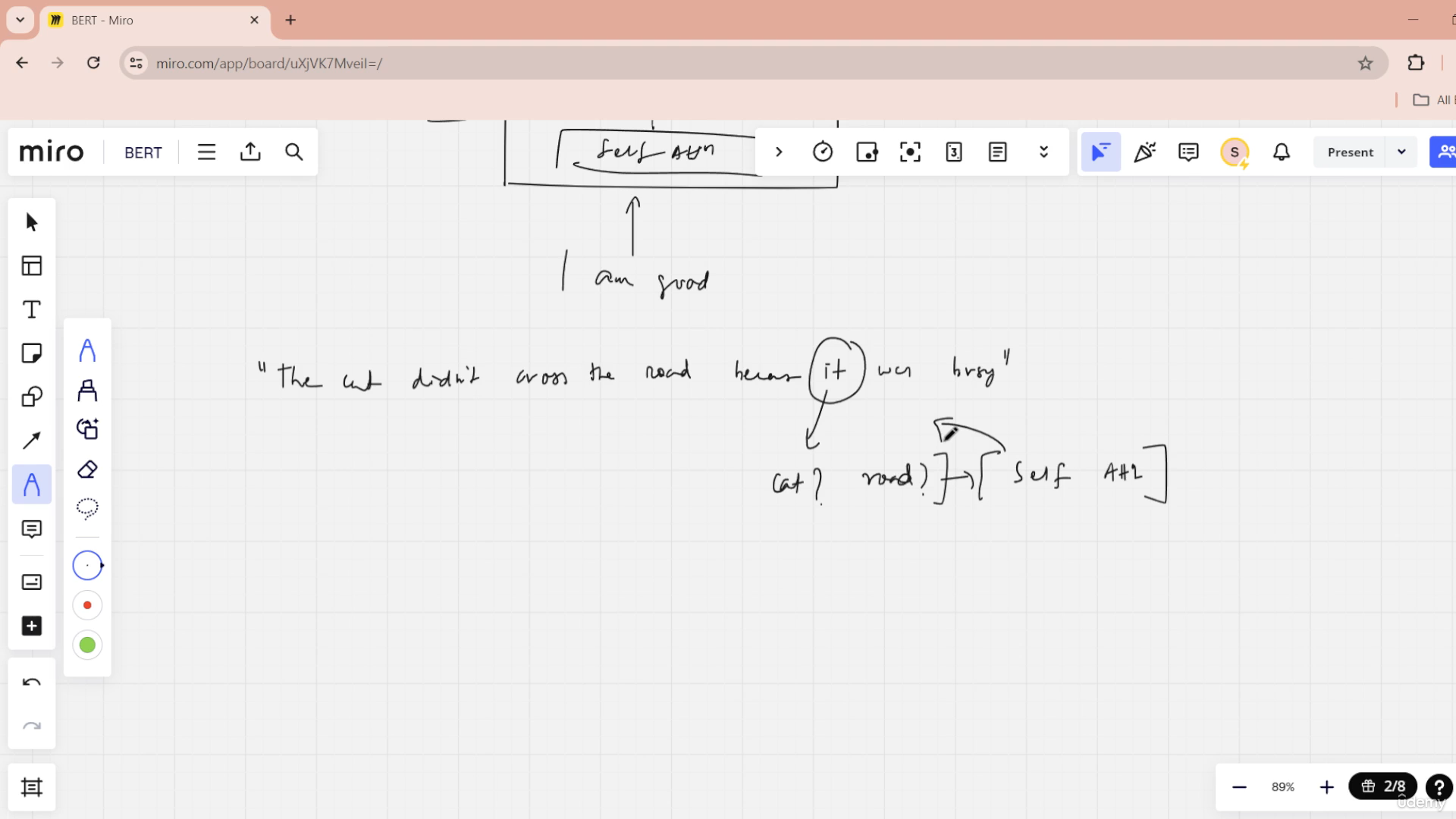
encoder, which uses multi-head self-attention

mechanisms to encode the input sequence.



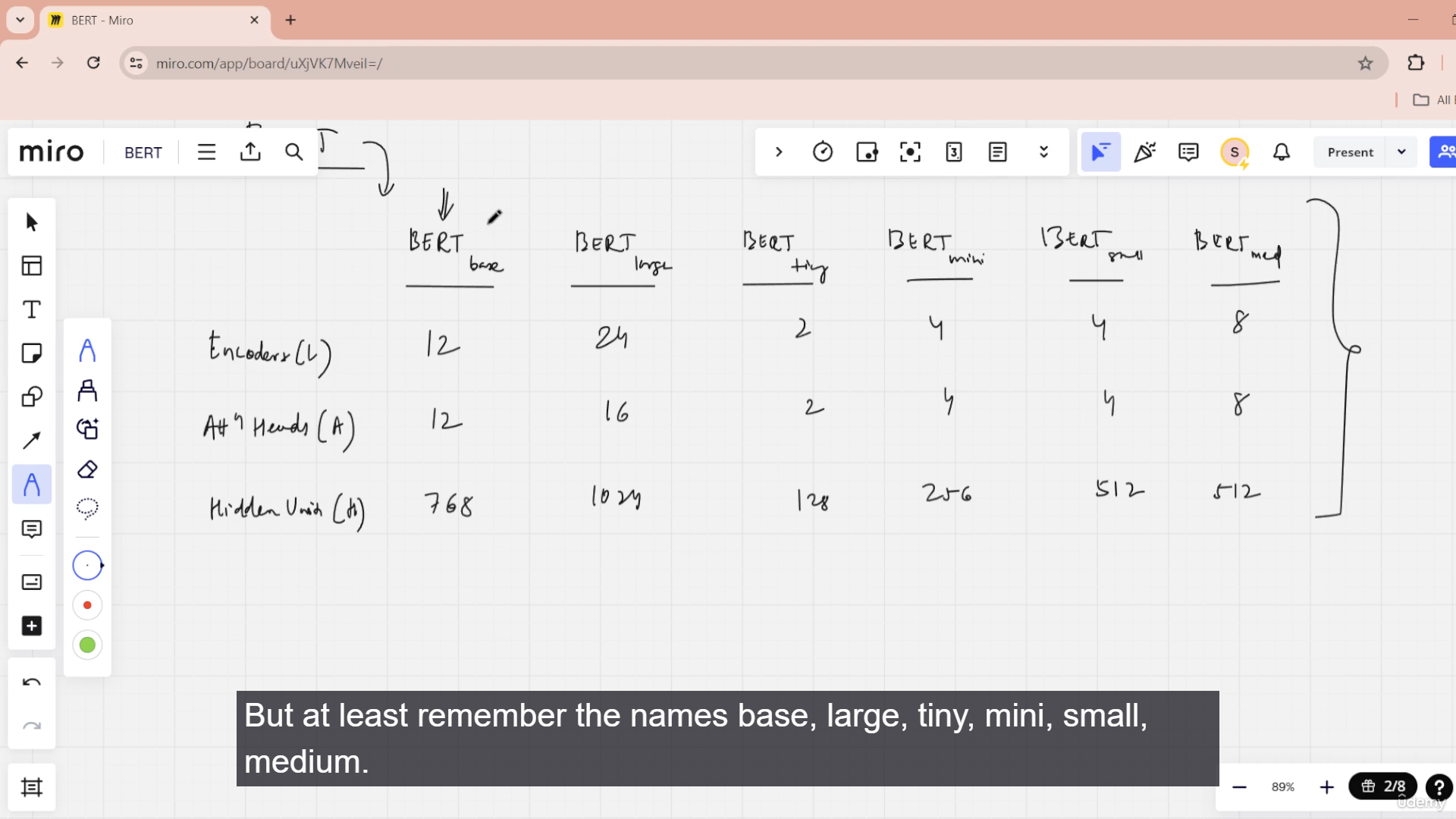






Introduction of BERT

BERT Configurations



Introduction of BERT

BERT Configurations

BERT Fine TUNING  
BERT PRE\_Training (MAsked LM)

INPUT Embedding BERT

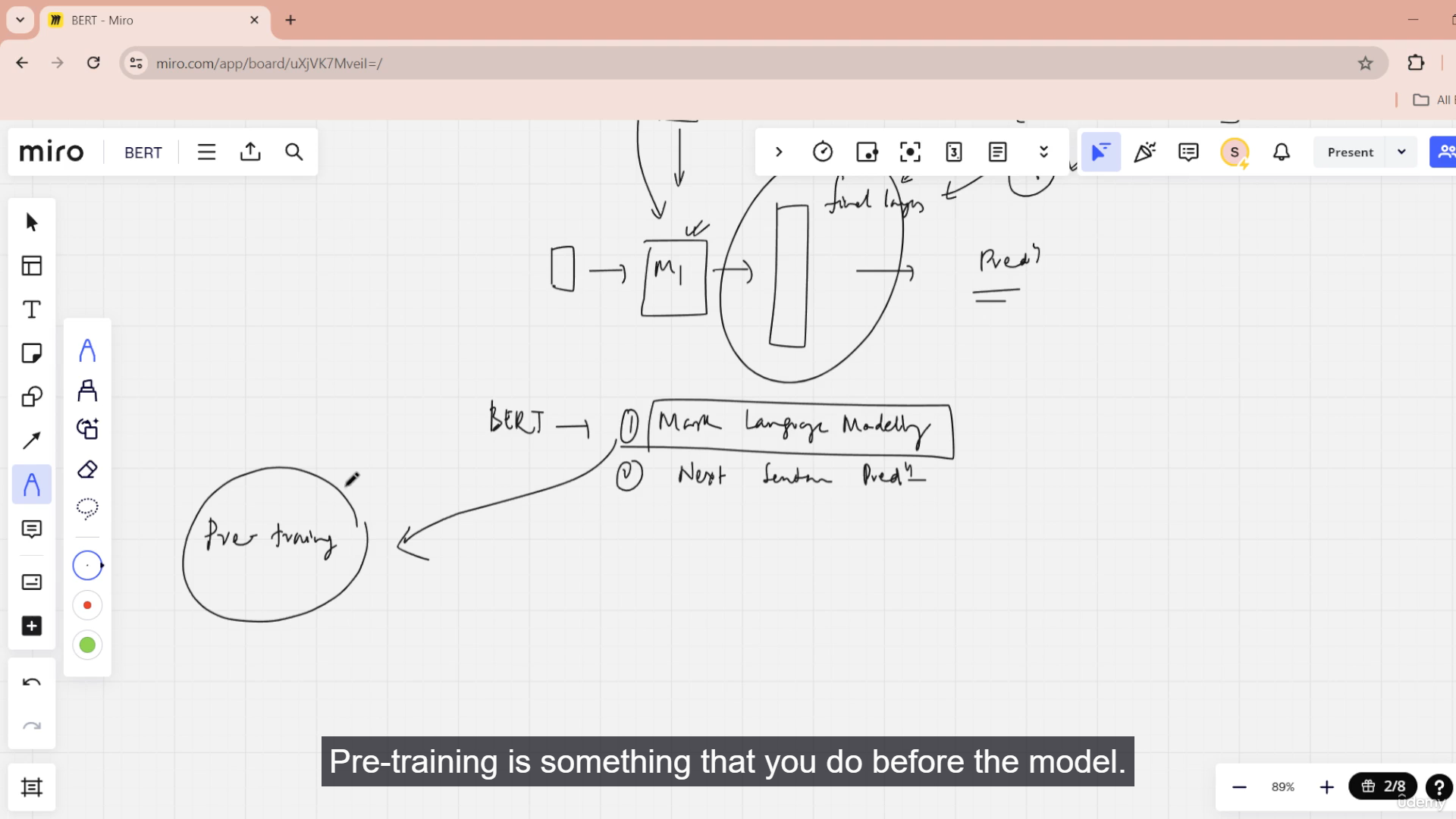
ARLM vs AELM

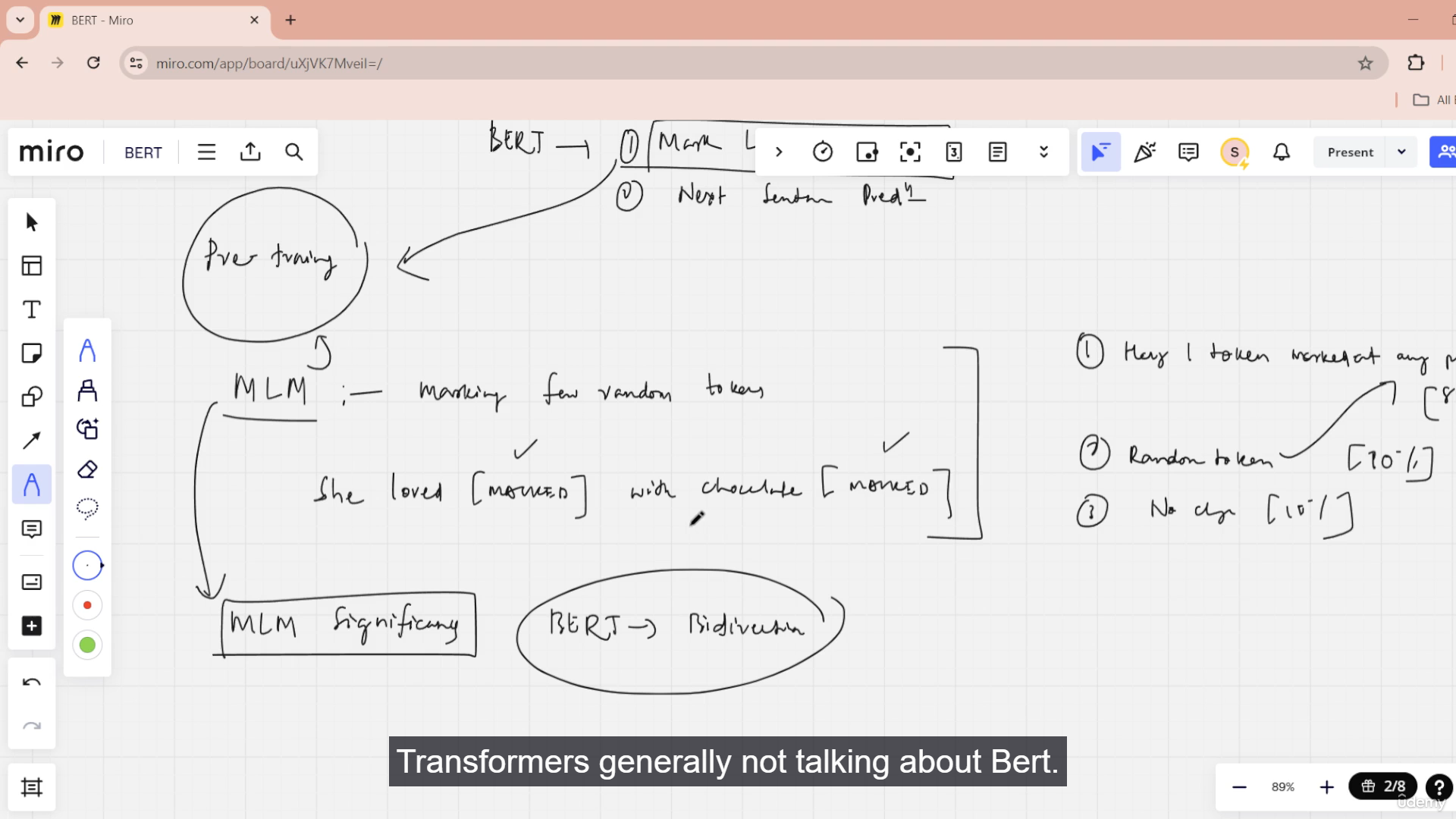
Roberta

distilBERT

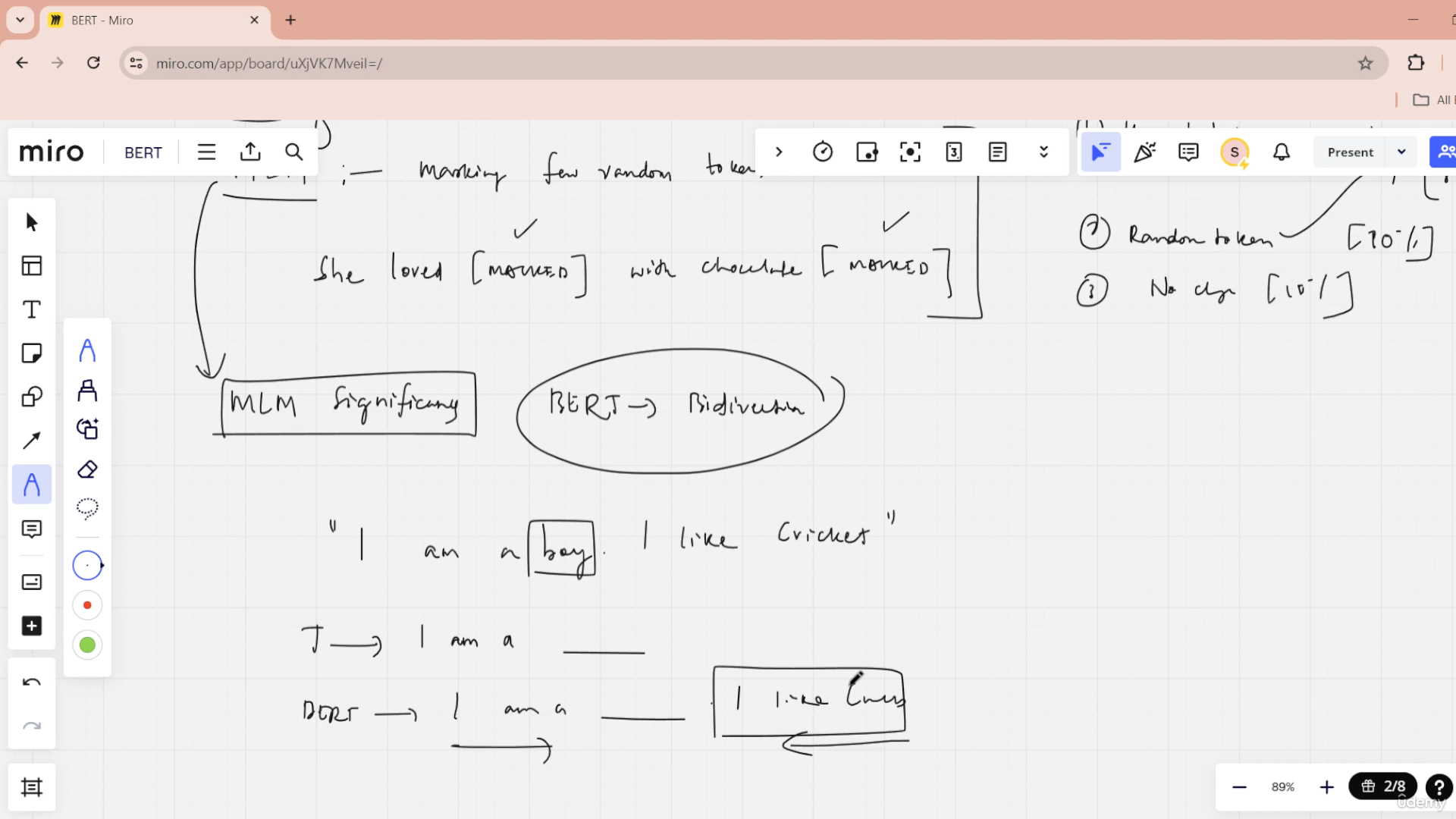
ALbert

BERT Fine TUNING

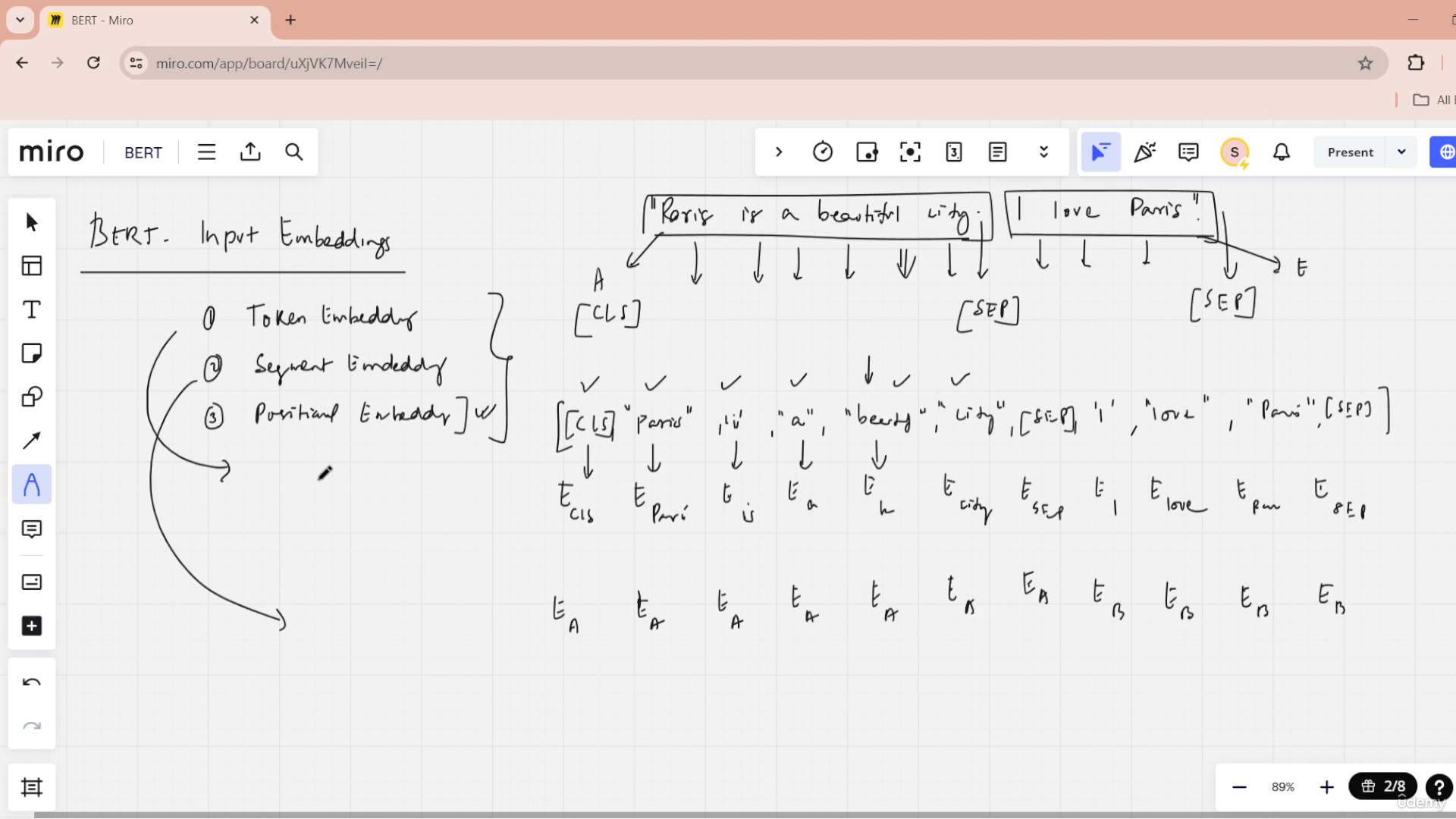


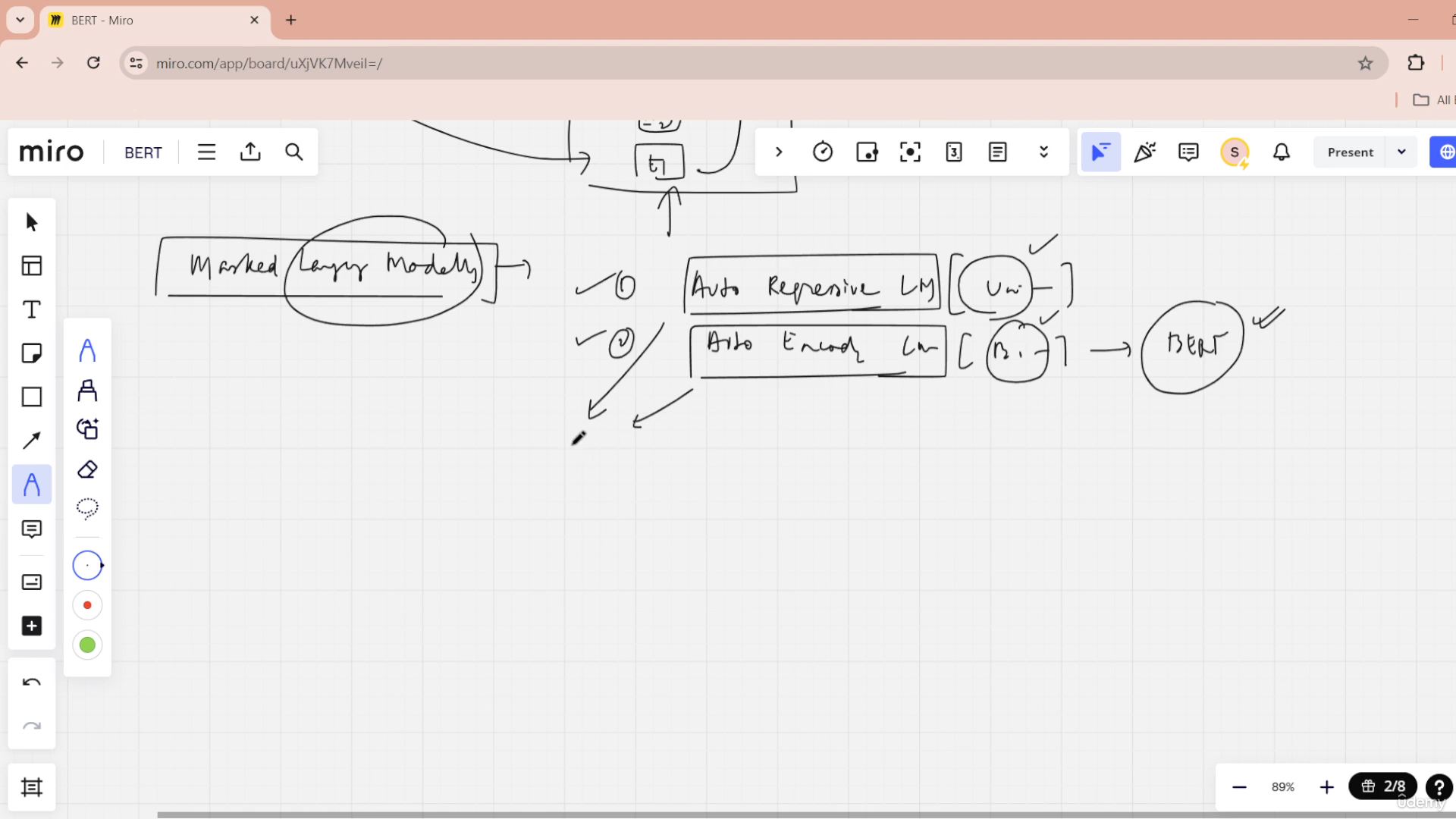


BERT PRE\_Training (MAsked LM)

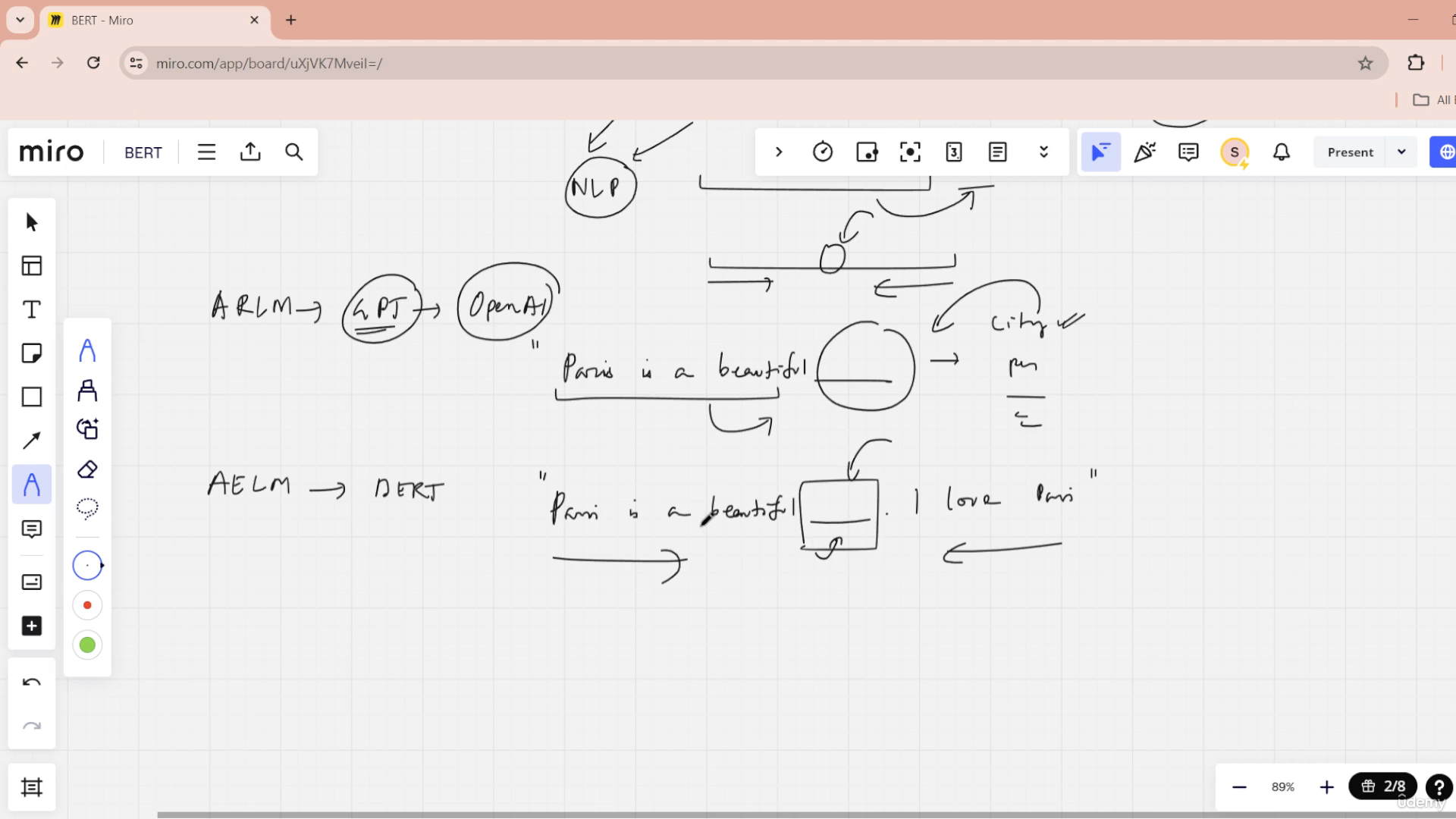


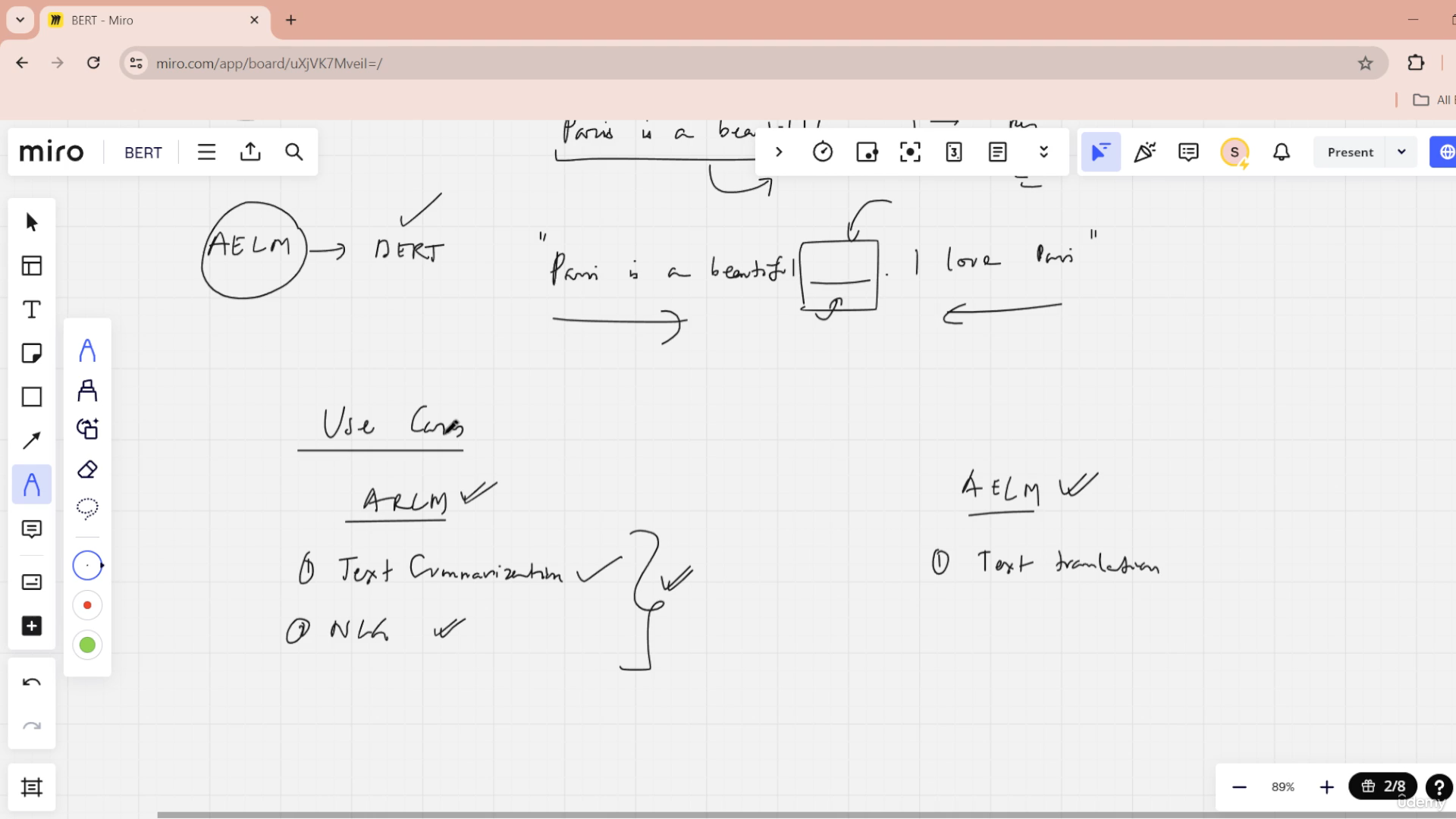
INPUT Embedding BERT





ARLM vs AELM





Roberta

distilBERT

ALbert

Role of Masked language modeling

MLM enables bidirectional learning by masking

(hiding) some words in the input text sequence.

BERT is trained to predict the masked words by

utilizing the context from the remaining words

both before and after the masked word.

This bidirectional context consideration allows

BERT to develop a deeper understanding of

language by leveraging information from both

directions.

BERT's objective is to accurately predict these

masked words based on the bidirectional

context.

© zepanalytics.com

Copyright © 2021 TalentLabs

Limited

Pre-training and fine-tuning BERT

Pre-training BERT involves two main tasks: Masked

Language Modeling (MLM) and Next Sentence Prediction

(NSP).

1. Masked Language Modeling (MLM)

● Input text is tokenized, and 15% of tokens are

randomly masked with [MASK].

● BERT is trained to predict the original masked tokens

from the remaining context tokens.

● This forces BERT to learn bidirectional representations

by leveraging context from both sides of the masked

tokens.

======

Next Sentence Prediction (NSP).

● Input sequences are formed by

concatenating two sentences, separated by

[SEP].

● BERT is trained to predict if the second

sentence follows the first in the original text.

● NSP allows BERT to learn sentence

relationships, beneficial for tasks like

summarization and QA.

Fine tuning BERT

We extend the pre-trained BERT with an additional output

layer and then fine-tune the resulting model's entire set of

parameters to train task-specific models.

Extending BERT with Task-Specific Layers

● Additional Output Layer: A new output layer is added to

the pre-trained BERT model to adapt it to the specific

task (e.g., classification layer for text classification tasks

or span prediction layers for QA tasks).

● Fine-Tuning Entire Model: Unlike freezing the pre-

trained layers, the entire set of parameters of the BERT

model, along with the new task-specific layers, is fine-

tuned. This means all weights are updated during

training on the task-specific dataset.

Let’s start with a **complete, clear explanation of BERT**, step-by-step 👇

## **🧠 What is BERT?**

**BERT** stands for **Bidirectional Encoder Representations from Transformers**.  
 It is a **Transformer-based** model developed by **Google AI (2018)** for **Natural Language Understanding (NLU)** tasks.

Unlike earlier models that read text **left-to-right** or **right-to-left**,  
 👉 **BERT reads the entire sentence in both directions simultaneously (bidirectional)**.

## **⚙️ Key Idea Behind BERT**

Traditional language models like GPT (before fine-tuning) predict the next word:

"The cat sat on the \_\_\_" → predict “mat”

But BERT does something different:

* It **masks random words** and **predicts them** based on both **left and right context**.
* This is called **Masked Language Modeling (MLM)**.

For example:

"The cat sat on the [MASK]"  
 BERT learns that the missing word is “mat” by understanding the **entire sentence context**.

## **🧩 Architecture of BERT**

BERT is built **entirely using the Encoder** part of the **Transformer architecture**.

### **🧱 Transformer Encoder Recap:**

* Uses **Self-Attention** → each word pays attention to all other words.
* Uses **Positional Encoding** → keeps track of word order.

### **📊 Common Variants:**

| **Model** | **Layers** | **Hidden Size** | **Parameters** |
| --- | --- | --- | --- |
| **BERT Base** | 12 | 768 | 110M |
| **BERT Large** | 24 | 1024 | 340M |

## **🧠 Two Training Objectives**

1. **Masked Language Modeling (MLM)** Randomly masks 15% of the words and predicts them.  
    Example:  
    Input → "I love [MASK] programming"  
    Output → predicts “Python”
2. **Next Sentence Prediction (NSP)** Trains BERT to understand relationships between two sentences.  
    Example:  
   * Sentence A: "I love playing cricket."
   * Sentence B: "It is my favorite sport."  
      BERT learns if B follows A (True) or not (False).

## **🧪 Fine-Tuning BERT**

Once pre-trained, BERT can be **fine-tuned** for various NLP tasks:

| **Task** | **Example** | **Fine-tuning Layer** |
| --- | --- | --- |
| **Text Classification** | Sentiment Analysis | Add Softmax layer |
| **Question Answering** | SQuAD dataset | Add span prediction head |
| **Named Entity Recognition (NER)** | Identify “New York” as LOCATION | Add token classifier |
| **Next Sentence Prediction** | Relationship between sentences | Binary classifier |

## **🧮 BERT Workflow Summary**

1️⃣ Input Text → Tokenization (WordPiece)

2️⃣ Add [CLS] at start, [SEP] between sentences

3️⃣ Convert to embeddings (word + position + segment)

4️⃣ Pass through Transformer Encoder (multi-head self-attention)

5️⃣ Take output from [CLS] → classification, or tokens → tagging

## **🧰 Example Code (Using Hugging Face Transformers)**

from transformers import BertTokenizer, BertModel

# Load pre-trained BERT

tokenizer = BertTokenizer.from\_pretrained('bert-base-uncased')

model = BertModel.from\_pretrained('bert-base-uncased')

# Example sentence

sentence = "BERT is a powerful transformer model."

# Tokenize

inputs = tokenizer(sentence, return\_tensors='pt')

# Get embeddings

outputs = model(\*\*inputs)

# CLS token representation

cls\_embedding = outputs.last\_hidden\_state[:, 0, :]

print(cls\_embedding.shape) # [1, 768]

## **🚀 Advantages**

✅ Understands context from both directions  
 ✅ Works well with small data after pretraining  
 ✅ Improves accuracy on many NLP tasks (e.g., SQuAD, GLUE)

## **⚠️ Limitations**

❌ Large model → needs GPU/TPU  
 ❌ Slow for long documents (>512 tokens)  
 ❌ Not suitable for text generation (use GPT instead)

## **🧩 Real-World Use Cases**

* Chatbots (intent classification)
* Customer sentiment analysis
* Search engines (Google uses BERT for understanding queries)
* Question Answering systems
* Information extraction from text

Would you like me to continue with:  
 1️⃣ **BERT fine-tuning example (e.g., Sentiment Analysis)** 2️⃣ **BERT architecture visualization (diagram + layer explanation)** 3️⃣ **Comparison with GPT and RoBERTa**

Which one should I show next?

Variants of BERT

1. RoBERTa (Robustly optimized BERT approach)

RoBERTa is an optimized version of BERT that emphasizes training with

more data and computational resources, as well as some modifications

in the training procedure.

Key Improvements:

● Training Data: Uses much larger training datasets.

● Training Duration: Trains for longer periods.

● Dynamic Masking: Changes masking patterns dynamically during

training instead of using static masking.

2. DistilBERT

DistilBERT is a smaller, faster, and cheaper version of BERT that uses

knowledge distillation to retain a significant portion of BERT's capabilities

while being lighter.

Key Improvements:

● Model Size: 40% fewer parameters than BERT.

● Speed: Runs 60% faster.

● Performance: Retains about 97% of BERT's performance on

language understanding benchmarks. © zepanalytics.com

Copyright © 2021 TalentLabs

Limited

3. ALBERT (A Lite BERT)

ALBERT is designed to be more parameter-efficient by sharing

parameters across layers and factorizing the embedding parameters,

leading to a smaller model with faster training times.

Key Improvements:

● Parameter Sharing: Shares parameters across layers to reduce

model size.

● Factorized Embedding Parameterization: Splits the embedding

matrix into smaller matrices to improve efficiency.

4. TinyBERT

TinyBERT is another distilled version of BERT, focusing on compressing

the model size while maintaining a good balance of performance.

Key Improvements:

● Model Compression: Applies both knowledge distillation and data

augmentation techniques.

● Size and Speed: Significantly smaller and faster than BERT.

© zepanalytics.com

Copyright © 2021 TalentLabs

Limited

5. SpanBERT

SpanBERT is an enhancement of BERT that focuses on better

capturing span-level representations, making it more

effective for tasks involving spans of text like question

answering and coreference resolution.

Key Improvements:

● Span Prediction: Trains the model to predict masked

spans of text instead of individual tokens.

● Objective Modification: Uses a span-boundary

objective to improve the model's ability to understand

the context of spans.