

# 05

## 实战演练之文本相似度

- (1) 文本匹配与文本相似度介绍
- (2) 基于Transfromers的解决方案
- (3) 代码实战演练（上，交互/单塔模型）
- (4) 代码实战演练（下，匹配/双塔模型）

# 抽奖活动

## 抽奖活动

- **活动奖品**

- AutoDL 充值 50 元
- 抽 2 人

- **参与方式**

- 本期视频 点赞+评论 即可参与抽奖

- **截止时间**

- 2023.07.21

**后续将作为长期的一个活动**

**创作激励每达500，取20%抽奖回馈给大家**

**希望各位小伙伴多多三连支持咯！**

# 文本匹配与文本相似度任务介绍

## 文本匹配与文本相似度简介

- **什么是文本匹配任务**

- 文本匹配 (Text Match) 是一个较为宽泛的概念，基本上**只要涉及到两段文本之间关系的，都可以被看作是一种文本匹配的任务，只是在具体的场景下，不同的任务对匹配二字的定义可能是存在差异的**，具体的任务场景包括文本相似度计算、问答匹配、对话匹配、文本推理等等，另外，如之前介绍的抽取式机器阅读理解和多项选择，本质上也都是文本匹配。
- 本次课程重点关注文本相似度任务，即判断两段文本是不是表达了同样的语义

# 文本匹配与文本相似度任务介绍

## 文本匹配与文本相似度简介

- 文本相似度

Sentence A	Sentence B	Label
找一部小时候的动画片	求一部小时候的动画片。谢了	1
别急呀，我的朋友。	你一定要看我一下。	0
明天多少度啊	明天气温多少度啊	1
可怕的事情终于发生。	你到底想说什么？	0

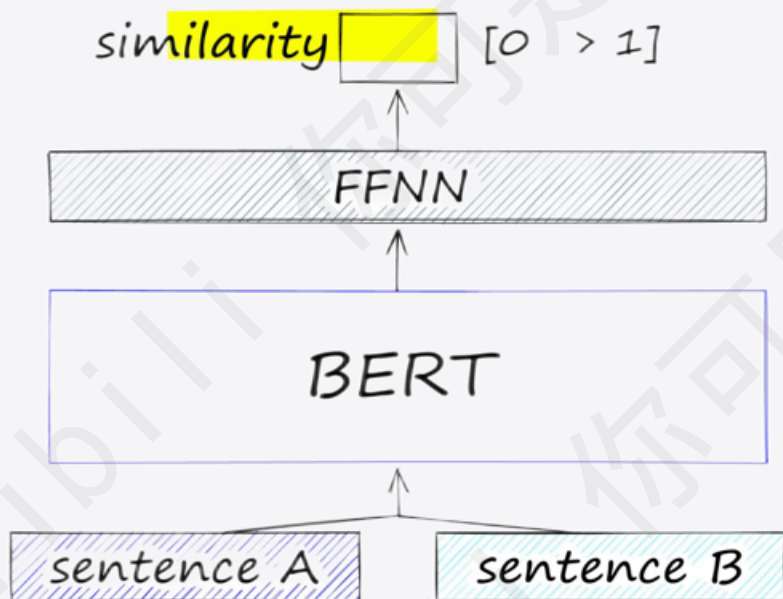
本质上是一项分类任务

# 基于Transformers的解决方案

## 基于Transformers的解决方案

- 最直观的解决方案

- 交互策略，输入句子对，对是否相似进行学习



# 基于Transformers的解决方案

## 基于Transformers的解决方案

- 最直观的解决方案

- 数据处理格式



- 模型训练方式



# 代码实战演练

## 代码实战演练（交互模式/单塔）

- 数据集

- simCLUE / train\_pair\_1w.json
- <https://github.com/CLUEbenchmark/SimCLUE/tree/main>

- 预训练模型

- hfl/chinese-macbert-base

# 基于Transformers的解决方案

## 基于Transformers的解决方案

- 最直观的解决方案

- 数据处理格式



- 模型训练方式



如何从多个候选文本中找出最相似的？



# 基于Transformers的解决方案

## 基于Transformers的解决方案

- 最直观的解决方案

- 数据处理格式



- 模型训练方式



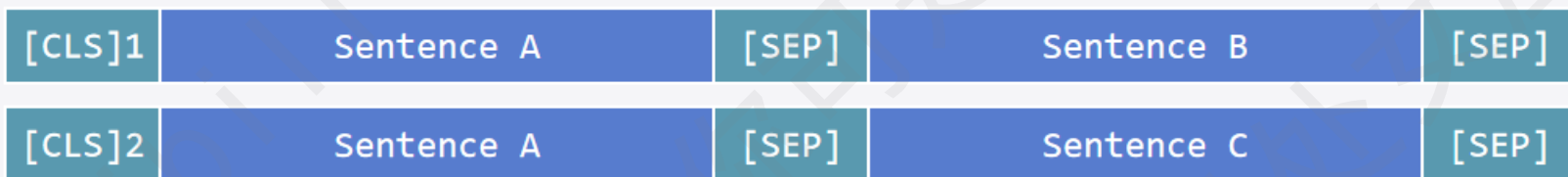
score    score > 0.5 label = 1  
          score < 0.5 label = 0

# 基于Transformers的解决方案

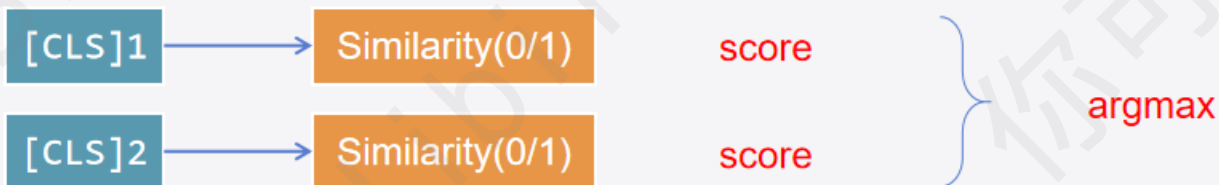
## 基于Transformers的解决方案

- 最直观的解决方案

- 数据处理格式



- 模型训练方式



# 代码实战演练

## 代码实战演练（交互模式/单塔）

- 数据集

- simCLUE / train\_pair\_1w.json
- <https://github.com/CLUEbenchmark/SimCLUE/tree/main>

- 预训练模型

- hfl/chinese-macbert-base

# 基于Transformers的解决方案

## 基于Transformers的解决方案

- 再看基于交互策略解决方案

- 效率问题

- 1个待匹配文本，1 000 000候选文本
    - 假设1次推理匹配 20ms
    - 1 000 000候选文本则需推理匹配1 000 000次
    - 消耗时间：20 000 000ms = 20 000s  $\approx$  333.33min  $\approx$  5.56h

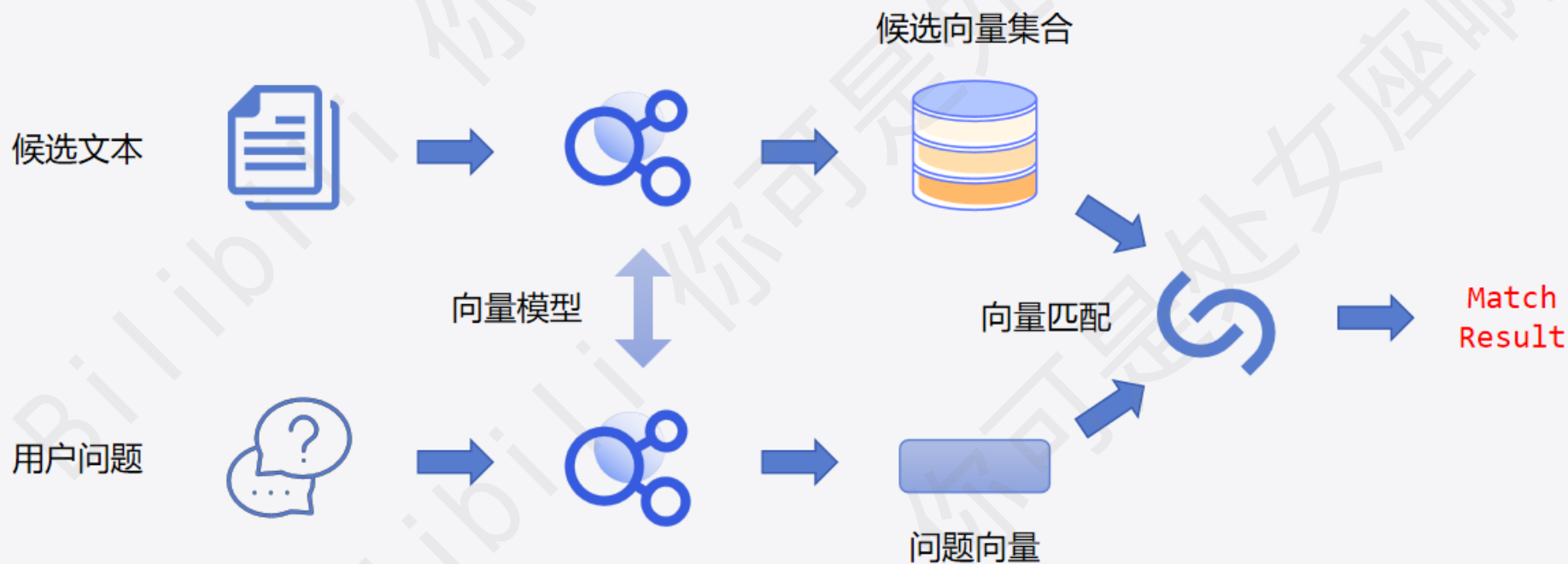
**效率极低，每次都需要与全量数据进行模型推理，数据量较**

**大时很难满足时延要求，如何解决？**

# 基于Transformers的解决方案

## 基于Transformers的解决方案

- 基于向量匹配的解决方案

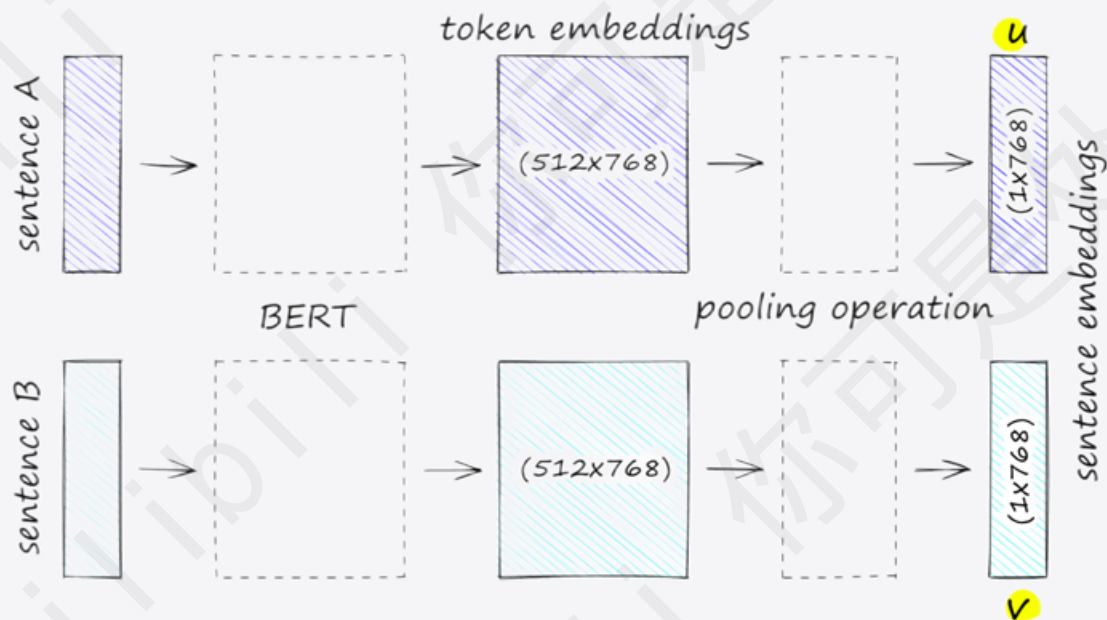


# 基于Transformers的解决方案

## 基于Transformers的解决方案

- 基于向量匹配的解决方案

- 向量匹配训练，分别对句子进行编码，目标是让两个相似句子的相似度分数尽可能接近1

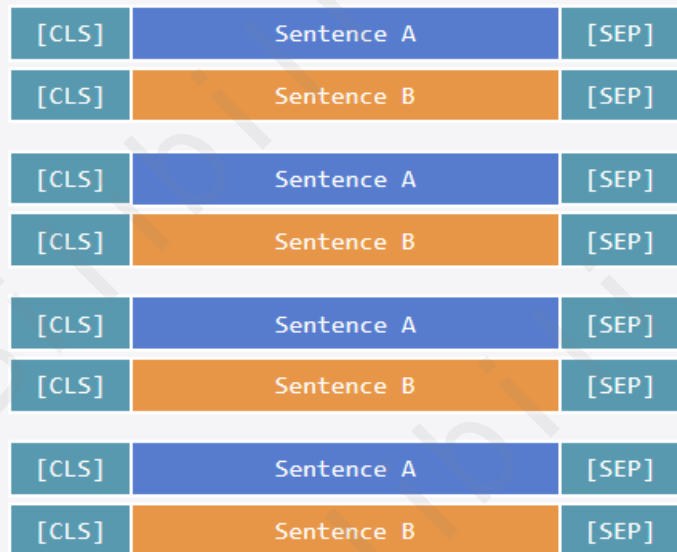


# 基于Transformers的解决方案

## 基于Transformers的解决方案

- 数据预处理

- 数据处理格式，与多项选择类似，因为要保证每个batch内都是对的数据



[CLS]	Sentence A	[SEP]
[CLS]	Sentence B	[SEP]
[CLS]	Sentence A	[SEP]
[CLS]	Sentence B	[SEP]
[CLS]	Sentence A	[SEP]
[CLS]	Sentence B	[SEP]
[CLS]	Sentence A	[SEP]
[CLS]	Sentence B	[SEP]

batch的数据维度

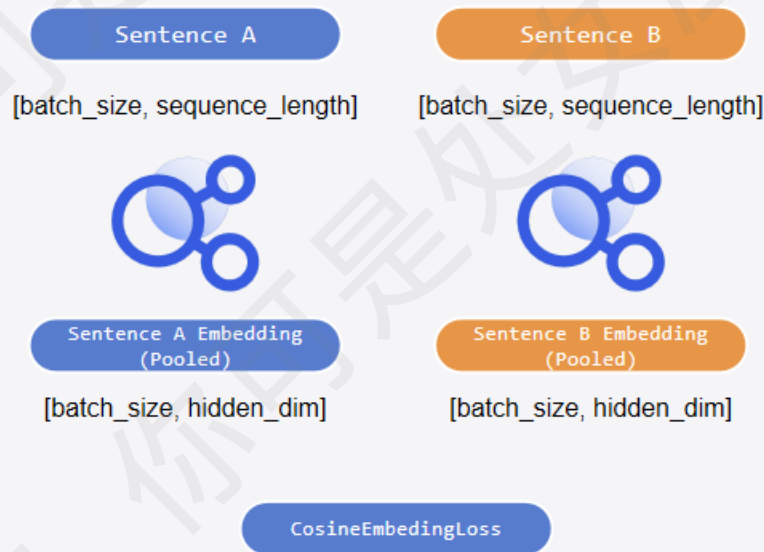
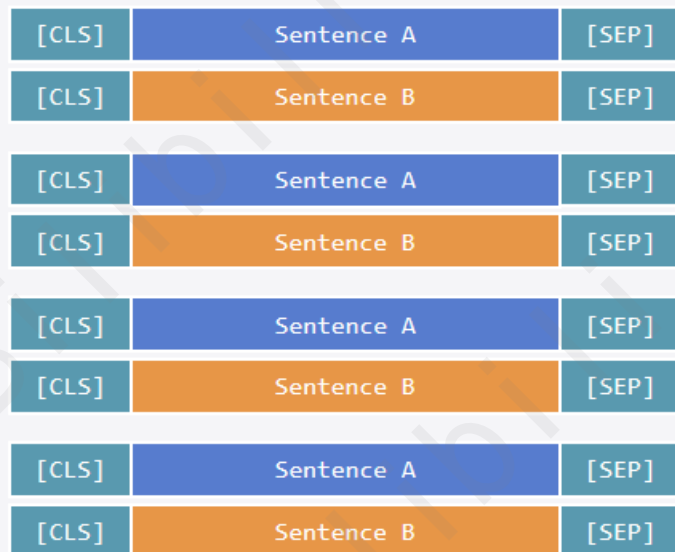
[batch\_size, 2, sequence\_length]

# 基于Transformers的解决方案

## 基于Transformers的解决方案

- 模型结构

- 没有预定义的模型，需要自行实现





# 基于Transformers的解决方案

## 基于Transformers的解决方案

- 模型结构

- CosineEmbeddingLoss

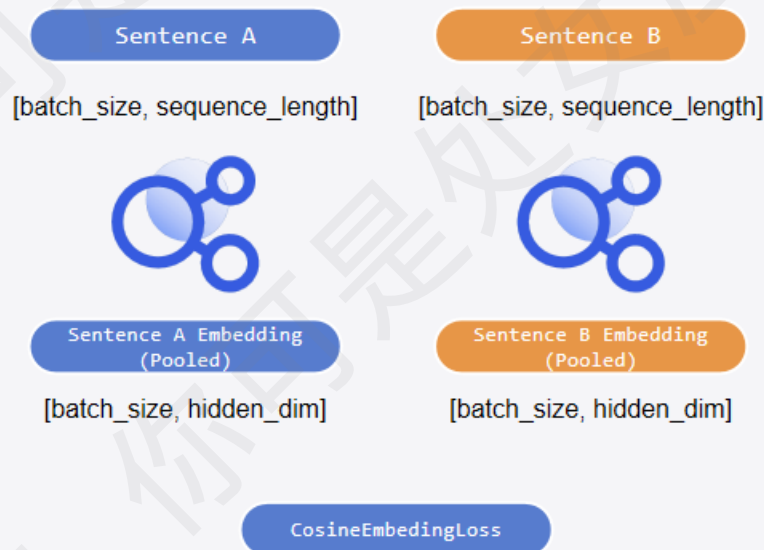
### COSINEEMBEDDINGLOSS

```
CLASS torch.nn.CosineEmbeddingLoss(margin=0.0, size_average=None, reduce=None,  
reduction='mean') [SOURCE]
```

Creates a criterion that measures the loss given input tensors  $x_1, x_2$  and a *Tensor* label  $y$  with values 1 or -1. This is used for measuring whether two inputs are similar or dissimilar, using the cosine similarity, and is typically used for learning nonlinear embeddings or semi-supervised learning.

The loss function for each sample is:

$$\text{loss}(x, y) = \begin{cases} 1 - \cos(x_1, x_2), & \text{if } y = 1 \\ \max(0, \cos(x_1, x_2) - \text{margin}), & \text{if } y = -1 \end{cases}$$



# 代码实战演练

## 代码实战演练（匹配模式/双塔）

- 数据集

- simCLUE / train\_pair\_1w.json
- <https://github.com/CLUEbenchmark/SimCLUE/tree/main>

- 预训练模型

- hfl/chinese-macbert-base

## 小结

- **文本相似度模型**

- 基于交互策略的单塔模型
- 基于向量匹配的双塔模型

- **更加便捷有效的工具**

- sentence-transformers <https://www.sbert.net/>
- text2vec <https://github.com/shibing624/text2vec>
- uniem <https://github.com/wangyuxinwhy/uniem>