基于对比学习的文本生成方法

姓名 安晨鑫
公司 复旦大学
职位 硕士研究生

目录
CONTENT

01 为什么要引入对比学习

对比学习在文本生成中到底可以扮演一个什么样的角色？

为什么在文本生成上应用对比学习

引入对比学习可以带来 1）从表示学习学习中获益，有价值的样本可以帮助模型学到更有意义的特征和表示 2）最近的研究[1] 表明对比学习可能是一个新的思路有助于缓解曝光偏差问题。

曝光偏差 (exposure bias)问题：指的是当前基于最大似然估计（MLE）训练的生成模型存在着测试和训练的不一致以至于损害模型的泛化性能。所谓曝光就是模型在训练阶段解码器只曝光给了正确的输入，而在测试阶段模型不得不基于他自己生成的字符来预测由此形成了测试和训练的偏差。

[1]Contrastive Learning with Adversarial Perturbations for Conditional Text Generation ICLR 2021

应用对比学习可以缓解自回归模型的曝光偏差问题

如何从对比学习的角度来尝试解决这一问题？

背景：对比学习的核心概念是要将正例拉近，负例推远。
正负样本的选择标准：以将质量高的人类写的目标为正例，那么另外一些包含错误的序列就作为负例。

缓解曝光偏差=> 将错误的样本和正确的样本同时在训练阶段曝光给解码器并利用对比学习的损失函数进行训练让模型去区别真实标签那些包含错误的序列。

一个简单的方法Naïve-CL

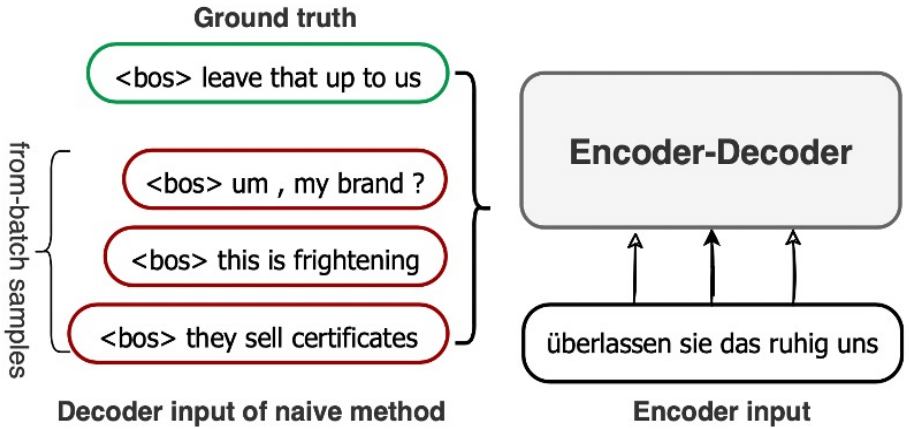
1. **正负例构建**：一个简单的方法是选择 SimCLR的方式，锚点 -- 编码器输入(source sequence), **正样本** -- 一般是人类写的标准目标语句（ground truth），**负样本** --同一个batch中随机选择其他的目标语句
2. **对比学习损失函数:** $InfoNCE\ loss \Rightarrow \mathcal{L}_{NCE}$

$$\mathcal{L}_{\text{NCE}} = -\log \frac{\exp(\cos(\mathbf{z}_{\boldsymbol{x}}, \mathbf{z}_{\boldsymbol{y}})/\tau)}{\sum_{\boldsymbol{y}' \in \mathcal{B}} \exp(\cos(\mathbf{z}_{\boldsymbol{x}}, \mathbf{z}_{\boldsymbol{y}'})/\tau)}$$

3. **训练目标** $\mathcal{L}_{NLL} + \mathcal{L}_{NCE}$

$$\mathcal{L}_{\text{NLL}} = -\sum_{t=1}^{N} \log p_\theta(y_t | \boldsymbol{x}, \boldsymbol{y}_{<t}).$$

4. **解码策略:** 普通的Beam search算法

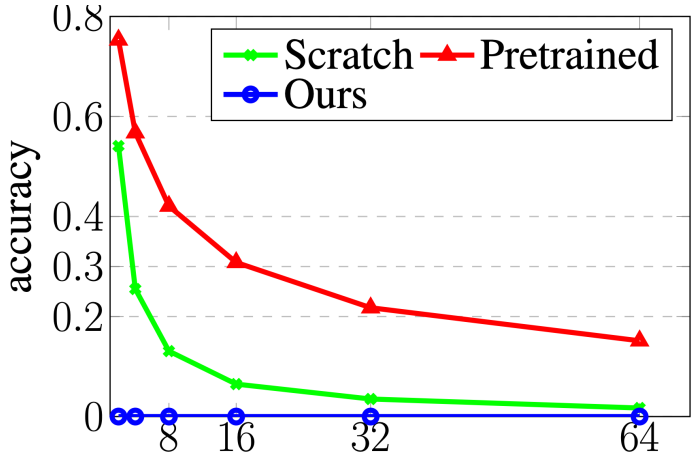


其他构造正负样本的方法

Naïve-CL 方法的一个明显的缺点是从同一个batch中随机选择的目标序列 非常容易区别 的

前人已经对做出了一些改进，这些改进主要涉及到扰动标准目标序列

1. **SSMBA（离散空间）**:在离散空间添加扰动，如随机mask一些词让一个用masked language model 将那些词预测回去生成新的正样本
2. **Dropout**：使用dropout机制类似于SimCSE，将ground truth输入进带有dropout机制的decoder两次所得到的不同表示为一对正样本
3. **CLAPS (目前效果最好的方法)**：在embedding空间对ground truth加扰动通过和原来的序列语义变化的大小作为划分正负样本的依据



Batch size 与选出ground truth准确率之间关系

目前基于对比学习的文本生成方法仍然存在瓶颈

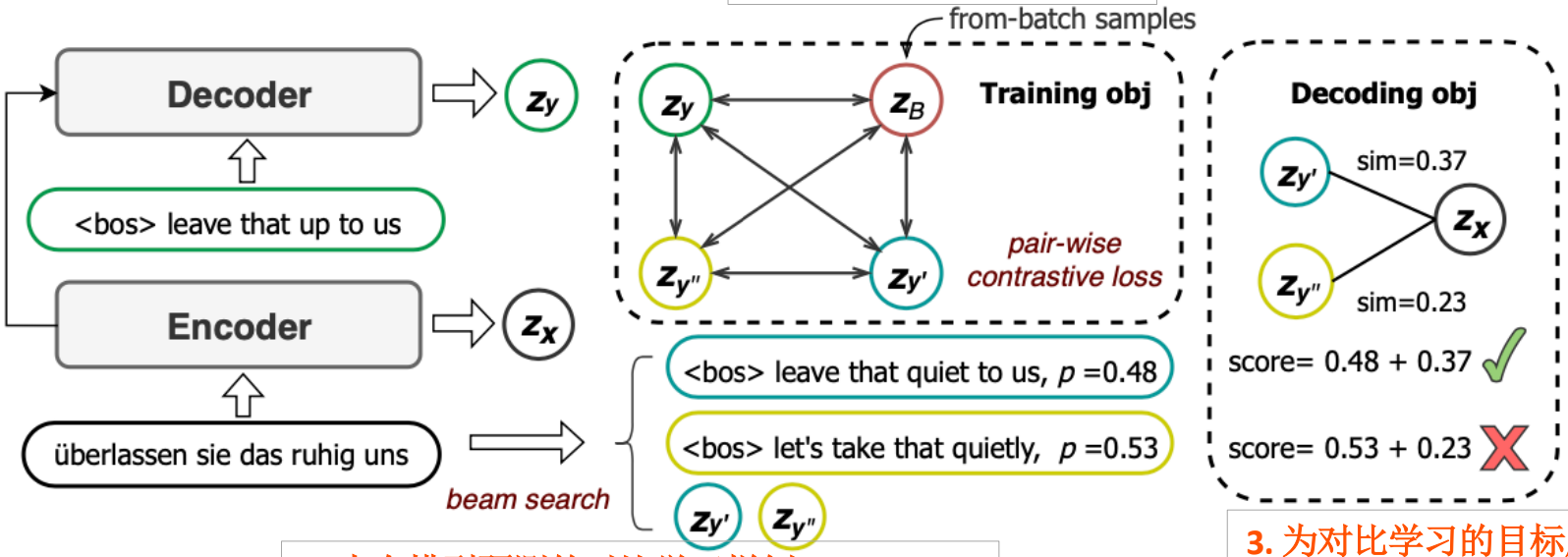
1) **正负例构建**：尽管之前的方法已经做出了一定的改进，但是对目标序列进行扰动并不能反映模型当前可能会出现的错误
2) **对比学习损失函数**：对比学习损失函数的选择也存在问题。InfoNCE Loss 只区分正负但是会忽略掉负样本之间的差异性
3) **解码目标：** 仅仅是简单的使用普通的beam search算法意味着这里存在着训练目标和解码目标的不一致

02 如何解决问题

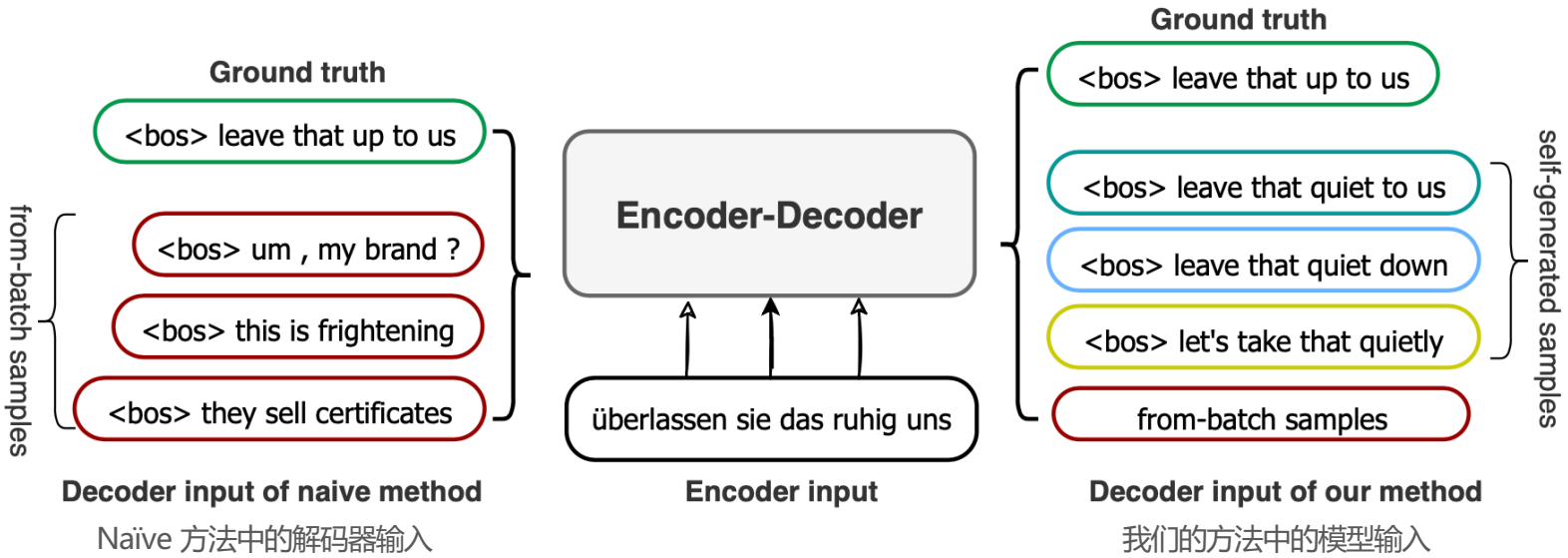
针对目前基于对比学习的框架存在的问题我们能做些什么改进？

我们的改进 – CoNT: **Co**ntrastive **N**eural **T**ext Generation

我们做了 3 件事...



我们的改进 – CoNT: **Co**ntrastive **N**eural **T**ext Generation



IWSLT14 De-En 翻译任务中的一个真实的例子

我们的改进 – CoNT: **Co**ntrastive **N**eural **T**ext Generation

1. 对比学习样例直接来自模型预测的结果：$y^{+,-} \sim p_\theta(y|x)$

2. 三元组的对比学习损失函数

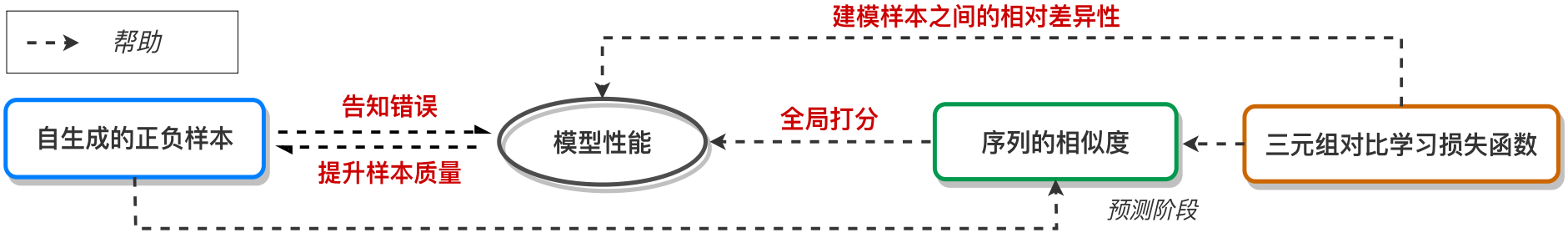
$$\mathcal{L}_{\text{N-Pairs}} = \sum_{(\boldsymbol{y}^+,\boldsymbol{y}^-)\in\mathcal{P}} \mathcal{L}(\boldsymbol{y}^+,\boldsymbol{y}^-) = \sum_{(\boldsymbol{y}^+,\boldsymbol{y}^-)\in\mathcal{P}} \max\{0, \cos(\mathbf{z}_{\boldsymbol{x}},\mathbf{z}_{\boldsymbol{y}^-}) - \cos(\mathbf{z}_{\boldsymbol{x}},\mathbf{z}_{\boldsymbol{y}^+}) + \xi\}$$

$\mathcal{P}$ 是包含 $\boldsymbol{k}$ 个对比学习 $\{y_1, y_2, \ldots, y_k\}$样本的pair集合 大小为 k(k-1)/2
对于每个(y_i, y_j)， + 和 - 是由他们各自的bleu score决定的 分数高的在这个piar中就为中另外一个就为负

3. 解码目标：语言模型打分(NLL建模) + 序列相似度(CL建模)

$$\boldsymbol{y}^* = \arg\max_{\hat{\boldsymbol{y}}}\{\alpha \cdot \cos(\mathbf{z}_{\boldsymbol{x}},\mathbf{z}_{\hat{\boldsymbol{y}}}) + (1-\alpha)\prod_{t=0}^{n} p(\hat{y}_t|\boldsymbol{x},\hat{\boldsymbol{y}}_{<t})\},$$

我们的总体设计



03 实验

经过我们加强过的对比学习框架是否足够强大了？

机器翻译 – IWSLT14 De-En, WMT16 Ro-En, WMT14 En-De

Model	WMT'16 Ro-En		IWSLT'14 De-En	WMT'14 En-De
	Tr-small	T5-small	Tr-small	Tr-base
MLE	25.78	28.21	34.18	27.30
Contrastive loss: InfoNCE loss				
Naive CL	25.49	27.79	34.45	27.28
SSMBA CL	25.98	28.48	34.32	27.16
Dropout CL	26.01	29.10	34.41	27.34
CLAPS†	23.59	29.41	–	–
CoNT	25.74	29.64	34.46	27.35
Contrastive loss: N-Pairs loss				
Naive CL	26.15	29.86	34.47	27.41
w/o seq sim	26.27	29.74	34.26	27.45
CoNT	**27.70**	**30.91**	**35.55**	**28.04**
w/o seq sim	27.42	30.54	34.69	27.77

使用不同正负样本构造方法所带来的差异

使用建模差异性的loss所带来的收益

文本摘要—XSum, Multi-News

Model	XSum			Multi-News		
	R-1	R-2	R-L	R-1	R-2	R-L
T5-small	36.10	14.72	29.16	42.36	15.34	21.91
T5-SSMBA CL	36.58	14.81	29.68	42.06	14.98	21.73
T5-Dropout CL	36.82	14.93	29.26	42.43	15.32	21.95
T5-CLAPS†	37.89	15.78	30.59	–	–	–
T5-Naive CL	36.34	14.81	29.41	42.20	15.18	21.78
T5-Naive CL (N-Pairs)	37.76	15.48	30.15	43.04	15.83	22.03
T5-CoNT	39.66	16.96	31.86	44.08	16.39	22.58
Earlier SOTA*	45.14	22.27	37.25	43.47	14.89	17.41
PEGASUS (base)*	39.79	16.58	31.70	42.24	13.27	21.44
PEGASUS (large)*	47.21	24.56	39.25	47.52	18.72	**24.91**
PEGA-CoNT	**47.76**	**24.69**	**39.46**	**48.68**	**19.29**	24.58

CoNT 大幅领先其他对比学习方法

Earlier SOTA 只在 pegasus large 之前最好的结果

代码注释生成Python Java 和 数据到本文生成WikiBio

Model	Python	Java
CodeBERT †	19.06	17.65
PLBART †	19.30	18.45
CodeT5 †	20.01	20.31
CodeT5-Dual-Gen†	20.11	20.41
With N-Pairs CL		
CodeT5-Naive CL	20.26	20.31
CodeT5-CoNT	20.43	20.56
With External Training Data		
CodeT5-Multi-Task†	20.36	20.46
REDCODER*	**21.01**	**22.94**

代码注释生成任务 给定一个函数输出它的作用

Model	BLEU
Table NLM †	34.70 ±0.36
vanilla Seq2Seq †	42.06 ±0.32
StructureAware †	44.89±0.33
R2D2 *	46.23±0.15
T5-small †	46.02±0.36
With N-Pairs CL	
T5-small-Naive CL	46.50±0.24
T5-small-CoNT	**47.17**±0.19

给定一个结构化的输入（infobox ，table，xml等） 输出其自然语言描述

数据到文本的生成--TOTTO

```
<page_title> List of Governors of South
Carolina </page_title> <section_title>
Governors under the Constitution of 1868
 </section_title> <table> <cell> 76 <
col_header> # </col_header> <col_header>
 74 </col_header> <col_header> 75 </
col_header> </cell> <cell> Daniel Henry
Chamberlain <col_header> Governor </
col_header> <row_header> 76 </row_header
> </cell> <cell> December 1, 1874 <
col_header> Took Office </col_header> <
row_header> 76 </row_header> </cell> </
table>
```

Sequence Output:

```
Daniel Henry Chamberlain was the 76th
Governor of South Carolina from 1874.
```

Model	Dev Set (All)		Dev Set (Non)		Test Set (All)		
	BLEU	PAR	BLEU	PAR	BLEU	PAR	BLEURT
BERT-to-BERT†	44.0	52.6	34.8	46.7	44.0	52.6	0.121
T5-large†	48.1	57.3	39.8	52.8	–	–	–
T5-3B†	48.4	57.8	40.4	53.3	**49.5**	58.4	0.230
T5-base†	47.7	57.1	39.6	52.6	–	–	–
T5-base-CoNT	**49.2**	**59.4**	**41.5**	**55.0**	49.1	**58.9**	**0.238**

和3B model取得on-par performance
仅仅使用220M的base model

常识生成 -- CommonGen

x_1 = { apple, bag, put }

y_1 = a girl puts an apple in her bag

常识生成 给定一些关键词 生成一个符合逻辑且通顺的句子

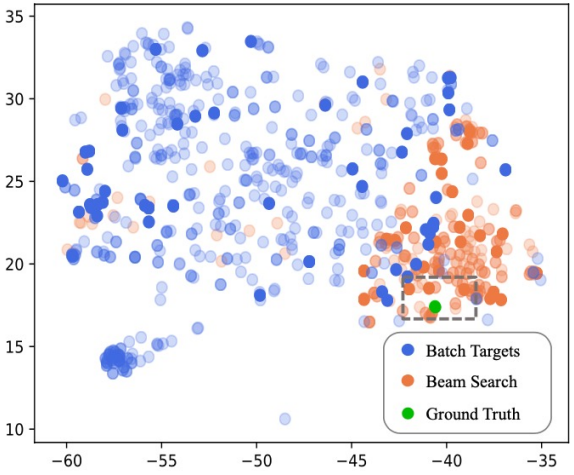
Model	ROUGE-2/L		BLEU-3/4		METEOR	CIDEr	SPICE	Coverage
GPT-2†	16.85	39.01	33.92	23.73	26.83	12.19	23.57	79.09
BART†	**22.02**	41.78	39.52	29.01	31.83	13.98	28.00	**97.35**
T5-large†	21.74	42.75	**43.01**	**31.96**	31.12	15.13	28.86	92.29
T5-base†	14.63	34.56	28.76	18.54	23.94	9.40	19.87	76.67
T5-base-CoNT	20.96	**43.15**	42.60	31.42	**32.05**	**15.96**	**28.95**	96.55
Human	36.72	54.45	52.55	46.49	38.79	37.64	52.43	99.33

所有的leaderboard 我们均只提交了一次

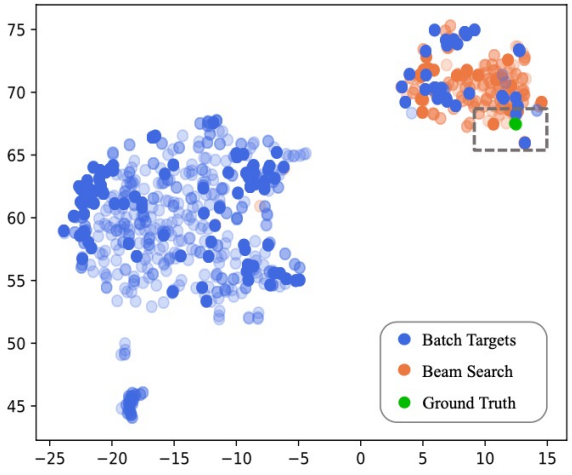
04 讨论

可能存在什么问题？

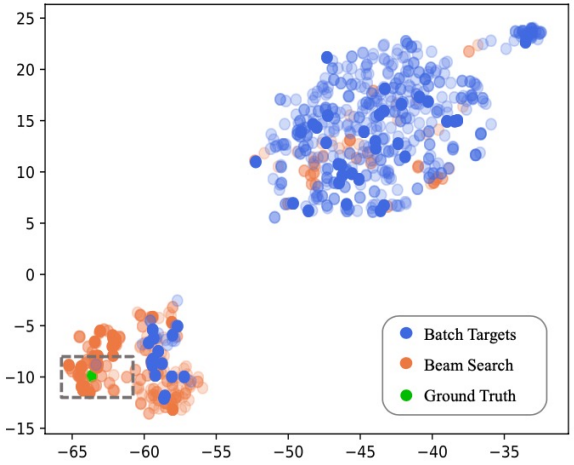
可视化表示 – 模型学到了什么样的表示



(a) MLE



(b) Naive CL



(c) CoNT

蓝色的点 代表 同一个batch中的样例，橘色代表是从模型分布中采样出来的 绿色表示ground truth的 颜色越深代表和ground truth越相似

序列相似度的权重 对模型最终性能的影响

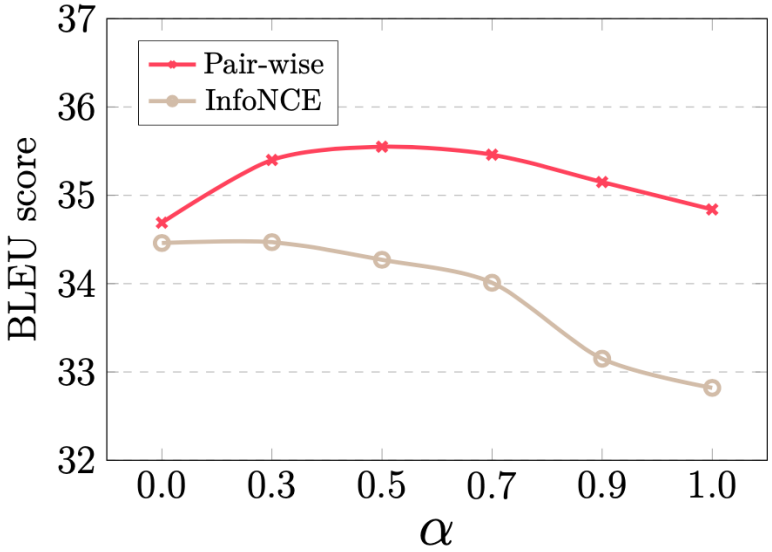


图1实验的设置：正负样本相同，使用不同的loss

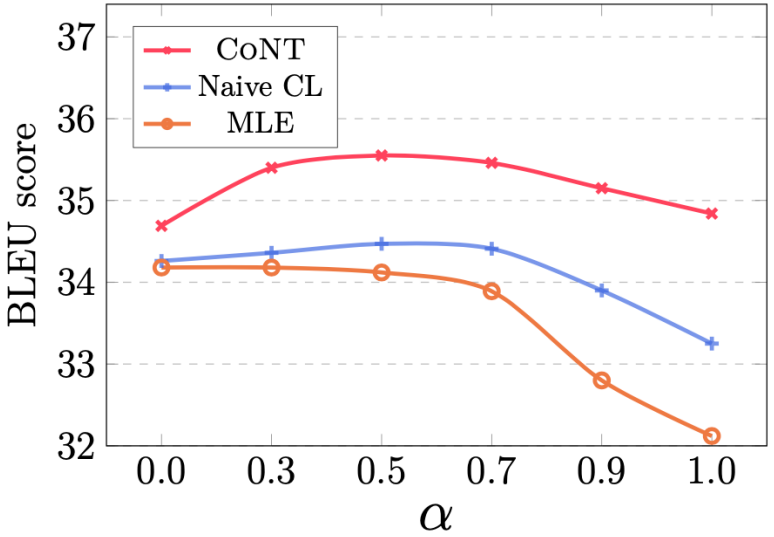


图2实验的设置：CoNT和naïve CL使用损失函数相同均为三元组对比。但是正负样本相同分别来自模型预测和batch。 MLE 为不做对比学习的baseline

如何在你的代码中使用对比学习

在训练阶段你需要在一个额外的loss

```
from CoNT import pair_loss
if nll_loss has converged:
    # pass your decoder, encoder output, and beam size
    ctr_loss = pair_loss(decoder, H_x, beam_size)
```

在预测阶段你需要略微修改一下之前的推理代码

procedure INFERENCE($\hat{G}$, $\boldsymbol{x}$)

$\mathbf{H}_{\boldsymbol{x}} \leftarrow \hat{f}(\boldsymbol{x})$, $b \leftarrow$ beam size, $\alpha \leftarrow$ balance factor $\in (0, 1)$

$\boldsymbol{y}^{1:b}, \mathbf{P}_{\boldsymbol{y}}^{1:b}, \mathbf{H}_{\boldsymbol{y}}^{1:b}$ = BEAMSEARCH($\hat{g}, \mathbf{H}_{\boldsymbol{x}}, b$) ▷ Get b candidates with beam search

$\mathbf{z}_{\boldsymbol{x}}, \mathbf{z}_{\boldsymbol{y}}^{1:b} \leftarrow \text{Avg}(\mathbf{H}_{\boldsymbol{x}}), \text{Avg}(\mathbf{H}_{\boldsymbol{y}}^{1:b})$ ▷ Avg$(\cdot)$ is an average pooling function

$\mathbf{D}_{\boldsymbol{y}}^{1:b} \leftarrow$ Cosine distance between $\mathbf{z}_{\boldsymbol{x}}$ and representation of hypotheses $\mathbf{z}_{\boldsymbol{y}}^{1:b}$

$\mathbf{P}_{\boldsymbol{y}}^{1:b} \leftarrow$ Likelihood of hypotheses returned by beam search

$k = \arg\max_{i=1..b}\{\alpha * \mathbf{D}_{\boldsymbol{y}}^{i} + (1-\alpha) * \mathbf{P}_{\boldsymbol{y}}^{i}\}$

return $\boldsymbol{y}^{k}$

CoNT的优缺点

在实际推理中，引入Contrastive learning几乎不会带来明显的浮点数运算操作（FLOPs）因此不会造成更多能量的消耗（不费电），并且我们和MLE框架下训练的模型推理时长几乎是一模一样的（不影响速度）

因此在实际部署中基于Contrastive learning训练的模型可以容易地替换现有的使用MLE 训练的模型

但是CoNT 的一个明显的缺点是：牺牲了训练的速度 CoNT的训练速度慢主要有三个方面

训练代码分析

Algorithm 2 Contrastive Text Generation: Given a generation dataset $<\mathcal{X}, \mathcal{Y}>$, a randomly initialized encoder-decoder model $\mathcal{M} = (f, g)$; return a contrastive generation model.

procedure WARMUP($\mathcal{M}$)
    Update the parameters of randomly initialized $\mathcal{M}$ with $\nabla_\theta \mathcal{L}_{nll}$ until convergence

procedure BEAMSEARCH(g, $\mathbf{H}_X$, b) ▷ beam search algorithm
    return Text, likelihood, logits of the b hypotheses

procedure TRAIN($\mathcal{M}$, $<\mathcal{X}, \mathcal{Y}>$)
    $\theta \leftarrow$ Parameters of $\mathcal{M}$, $b \leftarrow$ beam size
    WARMUP($\mathcal{M}$)
    while not convergence **do**
        $X^{1:k}, Y^{1:k} \leftarrow$ A minibatch of k datapoints from $<\mathcal{X}, \mathcal{Y}>$
        $\mathbf{H}_X^{1:k} \leftarrow f\left(X^{1:k}\right)$, $\mathbf{H}_Y^{1:k} \leftarrow g\left(\mathbf{H}_X^{1:k}, Y^{1:k}\right)$ ▷ outputs from the encoder and decoder
        $Y'^{1:k,1:b}, \mathbf{P}_{Y'}^{1:k,1:b}, \mathbf{H}_{Y'}^{1:k,1:b}$ = BEAMSEARCH$(g, \mathbf{H}_X^{1:k}, b)$
        $Y'^{1:k,1:(b+k)} \leftarrow$ Append b self-generated samples to $Y^{1:k}$
        for $i \in 1, 2, \ldots, k$ **do**
            for $j \in 1, 2, \ldots, , b + k$ **do**
                Do oracle measurement $o(Y'^{i,j}, Y^i)$ for each element $Y'^{i,j}$ in $Y'^{1:k,1:(b+k)}$
        $\mathcal{L}_{ctr} \leftarrow$ Get pair-wise contrastive loss
        update parameters using $\nabla_\theta(\mathcal{L}_{nll} + \mathcal{L}_{ctr})$
    return $\mathcal{M}$

训练代码分析

Algorithm 2 Contrastive Text Generation: Given a generation dataset $<\mathcal{X}, \mathcal{Y}>$, a randomly initialized encoder-decoder model $\mathcal{M} = (f, g)$; return a contrastive generation model.

procedure WARMUP($\mathcal{M}$)
    Update the parameters of randomly initialized $\mathcal{M}$ with $\nabla_\theta \mathcal{L}_{nll}$ until convergence

procedure BEAMSEARCH(g, $\mathbf{H}_X$, b) ▷ beam search algorithm
    return Text, likelihood, logits of the b hypotheses

procedure TRAIN($\mathcal{M}$, $<\mathcal{X}, \mathcal{Y}>$)
    $\theta \leftarrow$ Parameters of $\mathcal{M}$, $b \leftarrow$ beam size
    WARMUP($\mathcal{M}$)
    while not convergence **do**
        $X^{1:k}, Y^{1:k} \leftarrow$ A minibatch of k datapoints from $<\mathcal{X}, \mathcal{Y}>$
        $\mathbf{H}_X^{1:k} \leftarrow f\left(X^{1:k}\right)$, $\mathbf{H}_Y^{1:k} \leftarrow g\left(\mathbf{H}_X^{1:k}, Y^{1:k}\right)$ ▷ outputs from the encoder and decoder
        $Y'^{1:k,1:b}, \mathbf{P}_{Y'}^{1:k,1:b}, \mathbf{H}_{Y'}^{1:k,1:b}$ = BEAMSEARCH$(g, \mathbf{H}_X^{1:k}, b)$
        $Y'^{1:k,1:(b+k)} \leftarrow$ Append b self-generated samples to $Y^{1:k}$
        for $i \in 1, 2, \ldots, k$ **do**
            for $j \in 1, 2, \ldots, , b + k$ **do**
                Do oracle measurement $o(Y'^{i,j}, Y^i)$ for each element $Y'^{i,j}$ in $Y'^{1:k,1:(b+k)}$
        $\mathcal{L}_{ctr} \leftarrow$ Get pair-wise contrastive loss
        update parameters using $\nabla_\theta(\mathcal{L}_{nll} + \mathcal{L}_{ctr})$
    return $\mathcal{M}$

训练代码分析

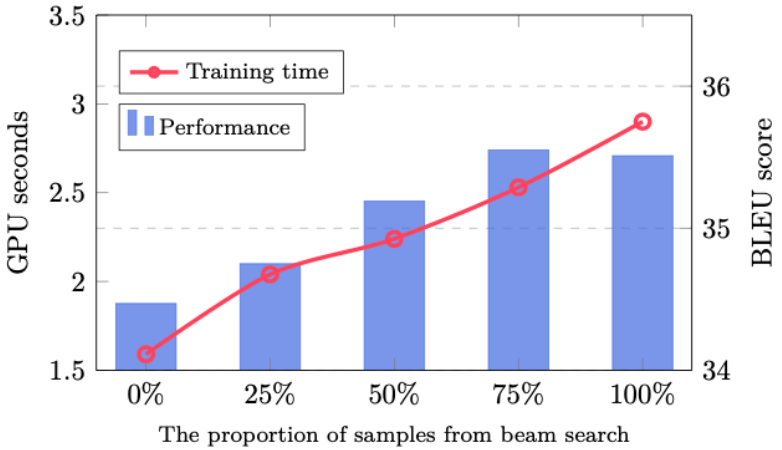
Algorithm 2 Contrastive Text Generation: Given a generation dataset $<\mathcal{X}, \mathcal{Y}>$, a randomly initialized encoder-decoder model $\mathcal{M} = (f, g)$; return a contrastive generation model.

procedure WARMUP($\mathcal{M}$)
    Update the parameters of randomly initialized $\mathcal{M}$ with $\nabla_\theta \mathcal{L}_{nll}$ until convergence

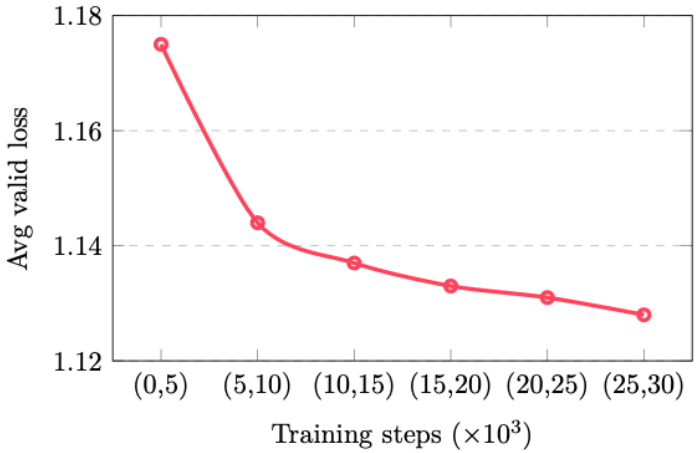
procedure BEAMSEARCH(g, $\mathbf{H}_X$, b) ▷ beam search algorithm
    return Text, likelihood, logits of the b hypotheses

procedure TRAIN($\mathcal{M}$, $<\mathcal{X}, \mathcal{Y}>$)
    $\theta \leftarrow$ Parameters of $\mathcal{M}$, $b \leftarrow$ beam size
    WARMUP($\mathcal{M}$)
    while not convergence **do**
        $X^{1:k}, Y^{1:k} \leftarrow$ A minibatch of k datapoints from $<\mathcal{X}, \mathcal{Y}>$
        $\mathbf{H}_X^{1:k} \leftarrow f\left(X^{1:k}\right)$, $\mathbf{H}_Y^{1:k} \leftarrow g\left(\mathbf{H}_X^{1:k}, Y^{1:k}\right)$ ▷ outputs from the encoder and decoder
        $Y'^{1:k,1:b}, \mathbf{P}_{Y'}^{1:k,1:b}, \mathbf{H}_{Y'}^{1:k,1:b}$ = BEAMSEARCH$(g, \mathbf{H}_X^{1:k}, b)$
        $Y'^{1:k,1:(b+k)} \leftarrow$ Append b self-generated samples to $Y^{1:k}$
        for $i \in 1, 2, \ldots, k$ **do**
            for $j \in 1, 2, \ldots, , b + k$ **do**
                Do oracle measurement $o(Y'^{i,j}, Y^i)$ for each element $Y'^{i,j}$ in $Y'^{1:k,1:(b+k)}$
        $\mathcal{L}_{ctr} \leftarrow$ Get pair-wise contrastive loss
        update parameters using $\nabla_\theta(\mathcal{L}_{nll} + \mathcal{L}_{ctr})$
    return $\mathcal{M}$

一些trade-off的方法



1. 减小样本中来自模型分布的样本数量增大batch中的样本数量



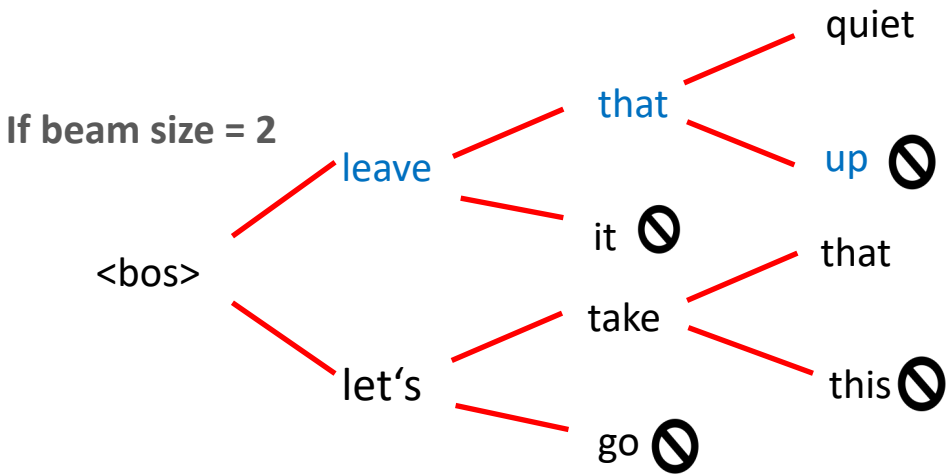
2. 在验证集中对比学习的下降曲线在前1w步比较陡 可以考虑early stop

利用序列的相似度进行协助解码

Ground truth: <bos> leave that up to us <eos>

Hypothesis1: <bos> leave that quiet to us <eos>

Hypothesis2: <bos> let‘s take that quietly <eos>



仅仅在beam search结束后再引入序列相似度作为额外的打分和postgeneration reranking有点相似

序列相似度其实并没有特别影响search的过程这可能不是最优的。

一个简单的方法考虑每生成K 步就计算和source的similarity 来决定beam的取舍

非常感谢您的观看