基于 Prompt 的小样本学习

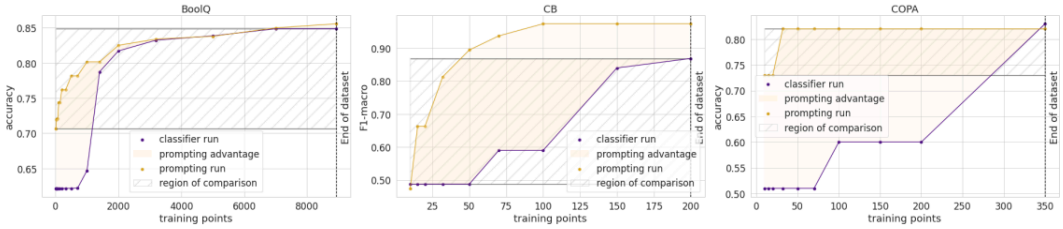
2022 年 1 月 4 日

1 实验简介

自 BERT、RoBERTa 等基于 Transformer 预训练的大模型在 NLP 各类任务上获得成功之后，在大量预料上现行进行预训练，再在下游任务数据集上进行调优 (Pretrain, Fine-tune) 的训练模式已经成为 NLP 模型的训练范式。语言模型能够通过预先提取预训练预料上的语义信息形成对语言任务的先验知识从而帮助我们在下游任务上调优到较好的性能。但是由于语言模型的预训练过程和下游任务之间仍然存在着一些差距，上述范式仍然需要有一定的下游标注数据量保证才能够达到较好的性能。

Prompt 方法正是针对这一问题而提出的，其不同于普通的模型对输入序列 x 通过估计 $P(y|x)$ 从而得到输出序列 y。Prompt 模型通过语言模型直接建模，对输入序列 x 使用一个模板处理得到 $x^{'}$ 来，利用掩码 (mask) 方式让语言模型对掩码位置的内容进行预测，以一种类似 “问答” 的形式从而得到最终的输出串 $\hat{x}$，利用输出串语义得到最终输出 y。这种范式又被称为 (Pretrain, Prompt, Predict)

Prompt 方法很好的弥合了预训练过程和下游任务之间的差距，从而减小了对下游任务数据量的依赖。经过实验，Prompt 方法可以在零次学习，小样本学习的情况下仍然取得优越的性能，远超过在相同数据量下调优的模型。



基于 Prompt 和基于 Fine-tune 模型对比 [1]

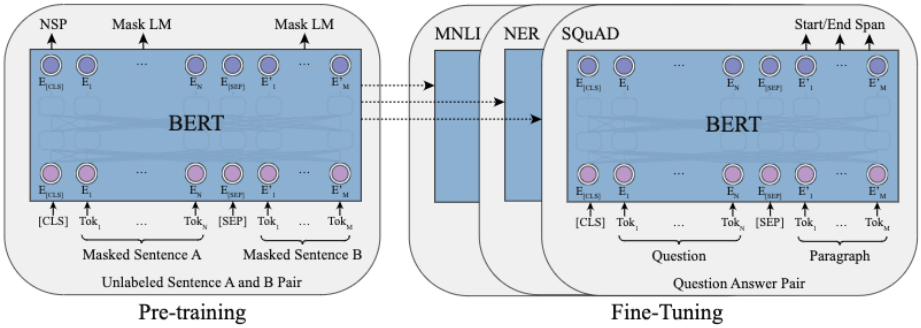
1.1 BERT 介绍

在之前的实验之中我们也接触过 BERT (Bidirectional Encoder Representations from Transformers)，即是双向的 Transformer 的 Encoder 模块，用于提取语言序列的语义信息。Transformer 中的 Attention 机制可以很好的解决此前序列化模型之中梯度消失问题，提供序列全局角度的特征提取，结合序列编码恢复序列的位置信息。能够很好的完成序列特征抽取工作。BERT 相比原先的 Transformer Encoder (6 层、2048 隐层大小) 更深更窄 (12 层，768 隐层大小)。在预训练过程中采用了掩码语言模型建立 (MLM) 和次句预测 (NSP) 两种方式，

提升语言模型对语句的理解能力。我们主要来讲一讲贴近 Prompt 的掩码语言模型预训练过程。

简单来说，掩码语言模型就是通过对输入序列以 15% 的概率用 [MASK] 替换，再训练模型预测出 [MASK] 位置原有内容的一种无监督训练模式，同时由于 [MASK] 并不会出现在下游任务的微调当中，BERT 采用一种策略即 80% 的情况下用 [MASK] 替换，10% 采用随机任意词，10% 保持不变的模型来尽量贴近下游任务。

以 BERT 模型为主，之后又产生了许多基于 BERT 的变体，例如 RoBERTa 和 AlBERT，其都是基于 BERT 模型之上，对 BERT 的预训练方法进行进一步的改进和调优，在此就不过多赘述。



BERT 模型

1.2 主流 Prompt 方法介绍

1.2.1 PET (Pattern Exploiting Training)[2]

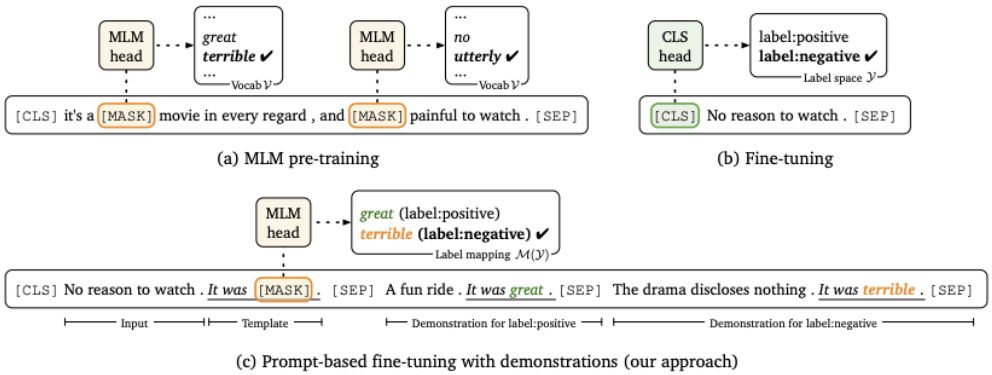
PET 模型的主要工作在于对 Prompt 模板的构建，采取一种半监督的学习方式，根据模板 (Pattern) 将输入序列映射为一个带 [MASK] 的完型填空文本类似 MLM 任务，让模型填空，进而 verbalizer 将模型的输出对应到相应的 label 之上，作者将其称为 PvP (Pattern Verbalizer Pair) 这也成为之后 Prompt 模型的范式。

同时为了对比多个不同 Pattern 的优劣，选出其中最好的一个，作者通过设计多个 PvP，多模型进行集成 (Ensemble) 给大量未标注数据打上软标签，进而分类器在打上软标签的数据上进行监督学习对软标签进行 finetune，类似知识蒸馏。同时作者为了使不同 prompt 模型能够相互学习提出了 iPET，其实就是利用半监督的方式，用之前的软标签打标签后拿增量的数据集继续训练模型，最后将多代模型集成。

PET 这篇文章可以说是正式开启了 Prompt 学习的范式，小样本学习上超越了原先基于预训练调优的方法。

1.2.2 LM-BFF[3]

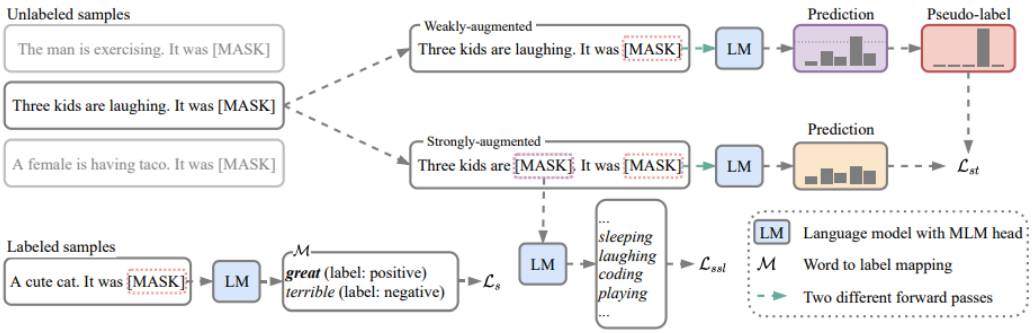
PET 中手工设计 Prompt 需要较多的经验，并且最后的效果并不一定是最优解。对此问题 LM-BFF 采用了自动 Prompt 的设计模式，对于模板和 Verbalizer 都进行了搜索。并且利用类似 GPT-3 的方法为 Prompt 添加示例 (demonstration)，以情感二分类任务为例，即添加正负例样本即其对应的 MLM 输出示例，来提示语言模型对应的输出形式从而进一步提升 Prompt 的效果。



1.2.3 SFLM[4]

最后是 2021 年 10 月 4 日最新的一篇论文 SFLM，其在 LM-BFF 的基础上提出了一种自监督的训练方法，即通过类似 MLM 的方式结合软标签模式，让模型能够更加注意到 Prompt 的含义。文章提出了弱增强方法即对未标注数据的普通 Prompt，在其上让模型预测一个软标签。结合强增强方法即在 Prompt 基础之上再对其做 MLM，按 15% 随机 MASK 掉位置之后，再进行预测，我们希望模型在对应 Prompt 掩码位置预测的软标签应当与之前弱增强方式一样，另外则是强增强位置应当能够预测出之前 MASK 的词语，因此在这两处添加两个损失。最后和普通 Prompt 的损失组合起来得到最终损失。$L = L_s + \lambda_1 L_{st} + \lambda_2 L_{ssl}$

其最终的结果也是超越了 PET 和 LM-BFF 达到了 SOTA



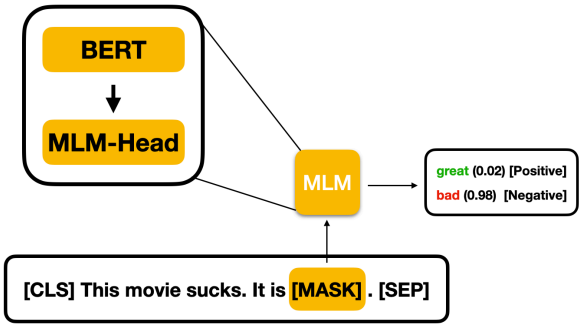
SFLM 架构

2 实验内容

在实验中我主要采用 RoBERTa-large 作为我的主要预训练模型，结合 BERT-large 和 BERT-base 进行对比实验。

2.1 零次学习 Zero-Shot

零次学习，即不依赖于训练样本直接生成结果。Prompt 方法能完成零次学习的原因是其任务模式和预训练模型的 MLM 预训练十分一致，实际上就是将下游任务通过 Prompt 建模为语言模型来完成，因此 Prompt 不需要在下游任务上进行训练也能有一定的准确率。



Prompt Zero Shot

在实验参数设置中，在 RoBERTa 中选用 Prompt 模板 <It is [MASK] . [SEP]>，Verbalizer 选用 great 和 bad 映射 positive 和 negative，即 $\mathcal{M}(t_{great}) = 1$，　$\mathcal{M}(t_{bad}) = 0$，和相应 BERT 模型对比，不进行训练直接生成结果 (zero-shot)，实验结果如下。

2.1.1 模型对比

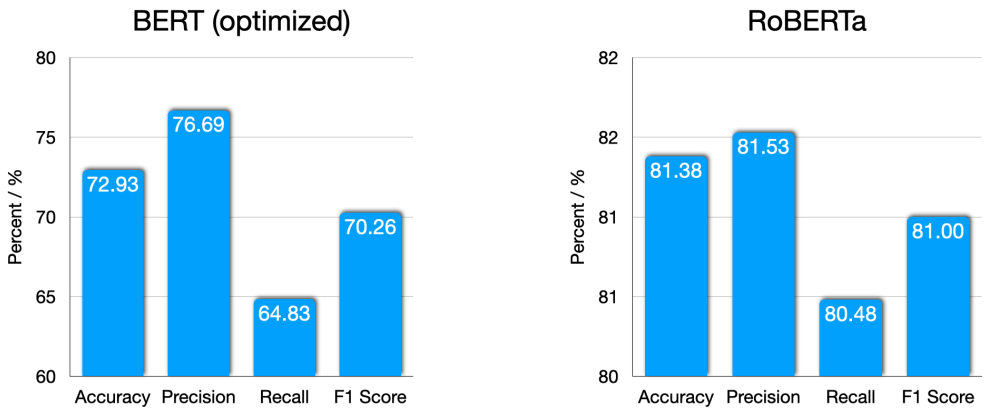


图 1: Prompt 零次学习，(1) 左图代表 BERT 模板经过人工筛选后得到的最好 Zero-Shot 效果，(2) 右图代表 RoBERTa 在上述模板配置之下的实验效果，两预训练模型均为 large 版本。

可以看到，基于 Prompt 的方法在零次学习中也可以达到较高的准确率，对模型来说，RoBERTa 预训练时间比 BERT 长，BatchSize 比 BERT 大，可以达到更好的效果。Prompt 模型 Zero-Shot 的准确率比较依赖于预训练的效果以及 Prompt 模板的设计。

2.1.2 同 “Fine-tune” 对比

为显示预训练模型 Zero-Shot 的效果，我进一步进行实验将其和直接预训练接分类头的模型进行对比。由于是 Zero-Shot，我们不能在下游任务数据集上进行 Fine-Tune，而是直接加载预训练输出结果。

经过实验，发现直接加载的预训练模型，由于未经过 fine-tune，其在下游任务上输出的预测均为 0，准确率为 0.5068，因为预测出来的 True Positive 为 0 因此 Precision 和 Recall 均为 0，即完全无法完成 Zero-Shot。

2.2 小样本学习 Few-Shot

Prompt 最主要的亮点在于其在小样本训练的条件下达到不逊色于大量数据集下调优的模型性能，这对于很多缺少标注的 NLP 任务是十分有价值的. 在 32/64/128 条标注的情况下，我在其上对 RoBERTa 模型开展利用 Prompt 和直接 Fine-Tune 结果的对比实验。采用的超参数和 LM-BFF 一文 [3] 中一致，即 lr=1e-5，batch_size=8，优化器采用 AdamW，训练 10 个 epoch 取其中性能最优。Fine_Tune 模型超参设置和 Prompt 一致，保证对比准确性。

2.2.1 模型性能对比

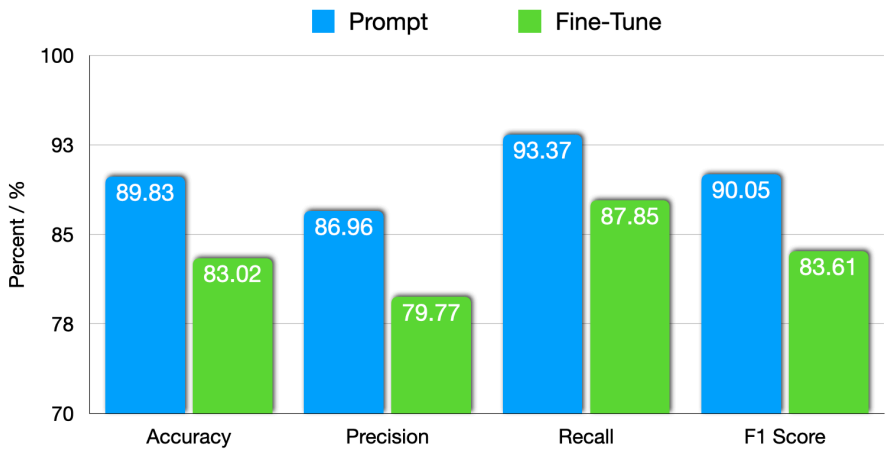


图 2: Prompt-FineTune 小样本学习对比 (128 条样本)

可以看出 Prompt 模型的准确率在小样本情况下要显著好于直接 Fine_Tune 的效果。我们进一步对比不同数据量下 (0/32/64/128) 两类模型的性能区别。

2.2.2 不同数据量下的性能对比

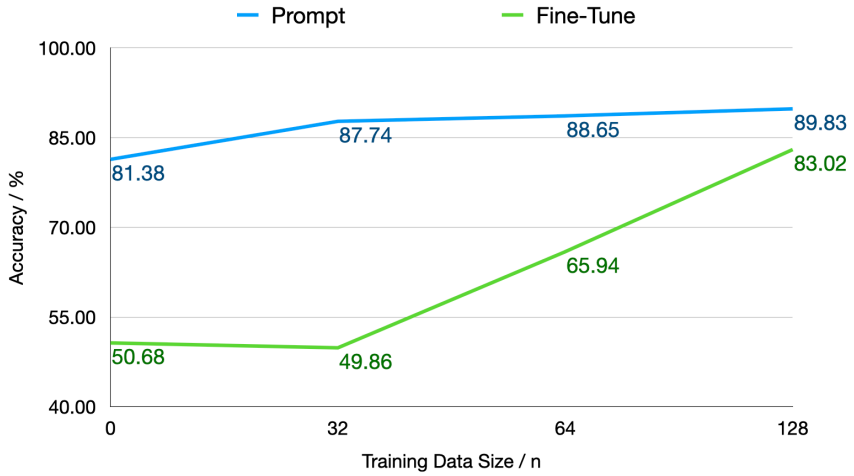


图 3: 不同数据量下 Prompt-FineTune 对比

可以看到 Prompt 在小数据量条件下依然发挥较好，仅仅 32 条数据就能非常好的拟合到 87.74% 的准确率，而对比之下，Fine-Tune 则非常依赖于数据量，可以看到在 32 条数据量训

练 10 个 epoch 的情况下，基于 Fine-Tune 模型的准确率不增反降，由于数据量较小其在训练集上过拟合严重，而根本无法泛化到 dev 集上，导致产生了极大的泛化误差。而当数据量增长之后，Fine-Tune 模型的准确率增长也较快，但是在小样本条件下仍然无法超越基于 Prompt 模型的准确率。可以看到普通深度学习范式的方法非常依赖于标注数据量，一旦数据量过小模型无法很好泛化，很容易出现在训练集上过拟合而在验证集上不拟合的情况。

2.3 模板设计 Template Design

Prompt 中较为重要的就是 Prompt 的模板设计，其关乎到我们如何告诉模型将下游任务建模为何种语言模型。模板的设计很大程度上关乎到 Prompt 模型的性能，但同时这也是非常困难的。对于 Prompt 模板的设计，目前基本上有两类工作，一类是通过人工手工进行设计，另一类是通过另外的语言模型进行搜索得出。

2.3.1 人工设计 Manuel

手工设计 Prompt 模板需要对于下游任务和模型的预训练有一定的了解才能设计出较好的模型。我们知道 BERT 模型的句子分割是利用 [SEP] 来进行标记的，因此在设计模板的时候，我们需要将 Prompt 的模板和原句连接再添加 [SEP]，而如果用 [SEP] 将原句和 Prompt 的句子进行切割分离再进行预测，则会影响模型的准确性，这是由 BERT 预训练的特性决定的, 因此我们将模板建立为如下形式。

$$X_i^{prompt} = X_i \ \circ \ It\ is\ [MASK]\ .\ [SEP]$$

其中 X_i 代表 i 条输入文本，$\circ$ 代表连接操作, X_i^{prompt} 代表第 i 条文本处理过后的 Prompt 模板形式。整个预测过程可以表示为如下内容。

$$P(y_i|x_i) = P([MASK] = \mathcal{M}(y_i|x_i^{prompt}))$$

在实验中，我进行多个模板的人工设计，固定 verbalizer 不变，采用 Zero-Shot 来避免训练过程中产生的训练误差，更好的体现模板的效果，最后结果如下表显示。

Template	Label Words	*Accuracy*/%
$< S_1 >$ It is [MASK] . [SEP]	great/bad	**81.38**
$< S_1 >$ It was [MASK] . [SEP]	great/bad	71.04
$< S_1 >$ it is [MASK] . [SEP]	great/bad	79.38
$< S_1 >$, which is [MASK] . [SEP]	great/bad	78.20
$< S_1 >$ That is [MASK] . [SEP]	great/bad	78.75
$< S_1 >$ x y [MASK] . [SEP]	great/bad	50.86
$< S_1 >$ It is [MASK] [SEP]	great/bad	77.11
$< S_1 >$ [MASK] . [SEP]	great/bad	73.39
$< S_1 >$ It is [MASK] .	great/bad	78.38

从中我们发现 It is [MASK] . [SEP] 作为模板最好，能够让模型更好的输出 MASK 位置的语义信息。同时首字母大写和最后的句号十分重要，前者是提示这段文本是新的句子的

起始，后者提示了 Label words 的输出，一直了其余句子中端可能出现词语的概率。最后用 [SEP] 对句子进行分割。这些都能够提示 Prompt 模型的准确率。

同时在极端情况下，我还实验了模板完全是随机乱码 x y [MASK] 和不使用其他提示直接让模型预测 [MASK] 位置的词语，可以看到前者效果非常差，接近随机猜测，可以证明语言模型确实建立了对于上下文语义的提取。单独 [MASK] 达到了较为中庸的性能 (73.39%)，体现了 Prompt 模板对于模型性能的重要性和必要性。

2.4 标签映射 Verbalizer

Verbalizer 代表的是语言模型生成词到真实标签的映射关系 $[MASK] = \mathcal{M}(y_i|x_i^{prompt})$，语言模型生成词的选取与模板一样影响着 Prompt 模型的准确率。在实验中，我选取不同的生成词进行性能对比，来验证 Verbalizer 对于 Prompt 模型性能的影响。同样使用 Zero-Shot 进行测试，来避免训练带来的影响。

2.4.1 单标签映射

Template	Label Words	*Accuracy*/%
$< S_1 >$ It is [MASK] . [SEP]	great/bad	**81.38**
$< S_1 >$ It is [MASK] . [SEP]	good/bad	64.85
$< S_1 >$ It is [MASK] . [SEP]	nice/bad	67.20
$< S_1 >$ It is [MASK] . [SEP]	wonderful/terrible	**80.74**
$< S_1 >$ It is [MASK] . [SEP]	great/terrible	69.39
$< S_1 >$ It is [MASK] . [SEP]	irresistible/pathetic	**80.84**
$< S_1 >$ It is [MASK] . [SEP]	cat/dog	49.87
$< S_1 >$ It is [MASK] . [SEP]	dog/cat	50.14
$< S_1 >$ It is [MASK] . [SEP]	bad/great	18.62

可以看出和情感相关的 Verbalizer 进行正确的情感极性映射可以使模型达到较好的性能，而采取普通词语，例如 cat 和 dog 这类词，则无法很好的使语言模型进行情感建模。如果反转标签，在 Zero-shot 上的效果是最差的，因为其违背了原先模型预训练中学习到的词语语义。

在 LM-BFF[3] 文章中，看到其反转标签后进行训练也可以达到一定的准确率，这是因为训练时模型通过标注重新对于两个词语的语义信息进行了学习，反转了语义，但是其效果必然会比正确语义的标签来的差，这里我也进行了实验。

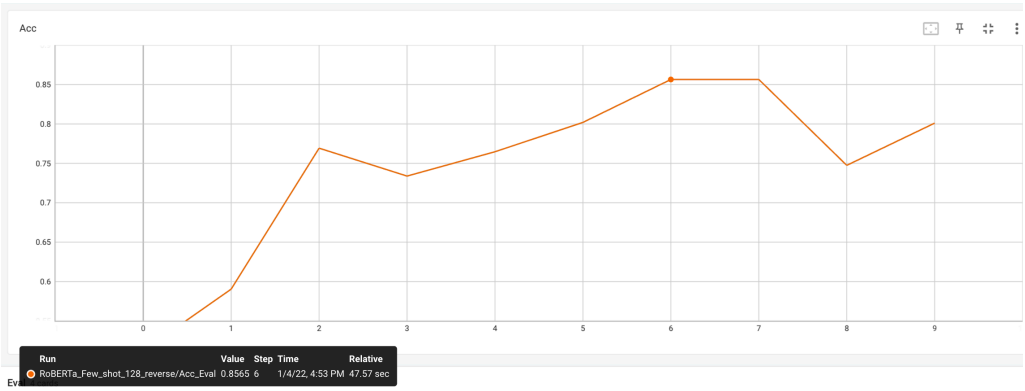


图 4: bad/great 标签反转训练 (128 条标注数据)

可以看到其仍然达到了 85.65% 的准确率，相比正确的 Verbalizer 降低了 4.18%，但值得注意的是，其仍然比在 128 条数据量上 FineTune 所得到的性能 (83.02%) 更高。

2.4.2 多标签映射

相对于单标签分类，我提出了多标签映射方法，即我们可以在训练时同时优化多组 label，将它们加权相加从而从权值判断属于 positive 还是 negative，这一任务可以建模为

$$\mathcal{M}(y_i|X_i) = \sum_{j=1}^{N} \alpha_i P(y_i|x_i^j), \quad X_i = \{x_1, x_2, \ldots, x_N\}$$

其中 N 代表多标签的组数, α_i 代表对应位置标签所占权重。同时我们在损失函数处同时优化多组标签，并将其损失函数加权相加组成最终的损失函数。

$$\mathcal{L}_{mll} = \sum_{i=1}^{N} \alpha_i \mathcal{L}_{CE}^{i}(y_{label}, y_{target})$$

在我的实验中我挑选在 2.4.1 节中表现最好的三组标签 great/bad, wonderful/terrible, irresistible/pathetic，进行多标签组合，这种做法类似于 ensemble，但是模型参数不变仅仅是修改了 Verbalizer 映射从一对一变为了多对一。我选取参数权重均等，即 $\alpha_i = 1/N$，所有超参数均和之前实验保持不变，实验结果如下。

Zero-Shot

Template	Label Words	*Accuracy*/%
$< S_1 >$ It is [MASK] . [SEP]	great/bad	81.38
$< S_1 >$ It is [MASK] . [SEP]	wonderful/terrible	80.74
$< S_1 >$ It is [MASK] . [SEP]	irresistible/pathetic	80.84
$< S_1 >$ It is [MASK] . [SEP]	Multi_Label (above all)	**84.20**

可以看到，经过多标签映射的 Prompt 模型，在 Zero-Shot 的情况下表现就十分突出，甚至直接超越了在 128 条标注数据上 Fine-Tune 的模型效果，可以看出多标签映射非常好的利用融合了预训练标签的语义信息，同时降低了模型对于标签选择的敏感度，增强了模型性能。

Few-Shot

我进一步进行了小样本下的模型训练，来进一步探索多标签映射 Prompt 模型的性能。可以看出，由于损失函数的设计，其在前期可以很好的帮助模型进行快速训练找到优化方向，但在后期多组损失函数之间呈拮抗关系，会影响模型的最终训练。这是由于语言模型输出 Verbalizer 的唯一性决定的，为此我设计损失函数根据 epoch 进行平滑，根据当前训练的 epoch 大小调整损失权重，当训练 epoch 大于 $threshold$ 后损失函数为 $\mathcal{L}_{CE}^{1}$，训练退化为普通单标签映射训练。

$$W_i = \frac{thre - epoch}{thre} * \alpha_i \ (i \neq 1) \quad W_1 = \frac{thre + (N-1) * epoch}{thre} * \alpha_1$$

最终在 128 条训练数据上达到了 89.5% 的性能。

可以看出多标签映射的情况其实仅仅只是优化了模型初始的优化空间，让其能够更加快速的收敛。而对模型性能的影响并不是很大。其主要意义是在于极小标注数据集下，尤其是 Zero-Shot 的情况，其能够非常大幅的提升性能。

2.5 模型对比 Model Comparison

在模型对比中我选择了两种模型，即 BERT、RoBERTa 之间的差异化对比。同时我也对 BERT-large 和 BERT-base 进行对比来探索相同模型参数量不同的情况下 Prompt 的效果差距。所有实验均在 128 条数据下进行，且保持超参数和 Promot 模板一致。

2.5.1 相同模型不同参数量对比

Model	*Accuracy*/%	*Precision*/%	*Recall*/%
BERT-base-uncased	79.56	78.70	80.29
BERT-large-uncased	82.47	79.88	87.29

相同模型，参数量上升准确率也有所上升，这符合我们的直觉，大模型能够更好的提取文本语义信息。

2.5.2 不同模型对比

Model [Large]	*Accuracy*/%	*Precision*/%	*Recall*/%
BERT	82.47	79.88	87.29
RoBERTa	89.83	86.96	93.37

在不同模型上，经过更久预训练的 RoBERTa 相比 BERT 有着更好的性能，符合我们的预期。

2.6 全量训练 Full data

最后我也对基于 RoBERT-large 的 Prompt 模型和普通 Fine-Tune 模型进行了全量训练，Prompt 模型达到 90.55%，普通 Fine-Tune 达到 91.24%。可以看到在全量训练的情况下，Fine-Tune 模型还是超越了 Prompt 模型，但二者差距并不大。

总结

在本次实验中，我们对 Prompt 模型进行了相对完整的学习，可以看到 Prompt 模型在小样本甚至零样本上的优势，对于很多 NLP 任务标注数据是很难获得的，因此 Prompt 模型在小样本上的出色表现具有实际价值。在实验中我们可以看到，Prompt 模型的性能十分依赖于模板和 Verbalizer 的选取，目前在大量文章 [3][4] 中已经出现自动搜索最合适的模板和 Verbalizer 的方法，这一方法进一步提升了 Prompt 模型的性能。同时 Promot 模型也非常依赖于预训练模型的准确率，如果预训练模型足够大且进行了充分的预训练，Prompt 的性能也会随之提升。

Prompt 能够成功的关键因素，在于其通过语言模型建模的方式缩小了预训练和下游任务之间的差距，让模型能够更好的适应下游任务，相比直接在下游任务上进行 Fine-Tune 需要更小的数据集，因此在数据量不足的情况下，考虑使用 Prompt 模型是非常好的选择。

参考文献

[1] Teven Le Scao, Alexander M. Rush. (2021). How Many Data Points is a Prompt Worth?.

[2] Timo Schick, Hinrich Schütze. (2021). Exploiting Cloze Questions for Few Shot Text Classification and Natural Language Inference.

[3] Tianyu Gao, Adam Fisch, Danqi Chen. (2021). Making Pre-trained Language Models Better Few-shot Learners.

[4] Yiming Chen, Yan Zhang, Chen Zhang, Grandee Lee, Ran Cheng, Haizhou Li. (2021). Revisiting Self-Training for Few-Shot Learning of Language Model.