Prompt is the technique of making better use of the knowledge from the pre-trained model by adding additional texts to the input.

* 目的：更好挖掘预训练语言模型的能力
* 手段：在输入端添加文本，即重新定义任务（task reformulation）

## NLP发展的四个范式：

* P1. 非神经网络时代的完全监督学习（Fully Supervised Learning, Non-Neural Network）
* P2. 基于神经网络的完全监督学习 (Fully Supervised Learning, Neural Network)
* P3. 预训练，精调范式 (Pre-train, Fine-tune)
* P4. 预训练，提示，预测范式（Pre-train, Prompt, Predict）

## 规律：每个范式都会涉及繁琐的，需要人来参与（提供先验）的工程（engineering）

P1: 特征工程（Feature Engineering）：2013年之前，使用传统的机器学习模型，需要手动定义输入文本的特征，如词性，长度等。Typical Work：CRF（Conditional Random Field）。

P2: 结构工程 （Architecture Engineering）：依赖神经网络，不需要手动处理特征，但是需要定义网络结构，如LSTM、CNN等。Typical Work：CNN for Text Classification。

P3: 目标函数挖掘 （Objective Engineering）：主要是fine-tuning。（有预训练语言模型，但与下游任务有gap）。Typical Work：BERT、ERNIE。

P4: Prompt挖掘工程（Prompt Engineering）：直接利用与训练语言模型辅以特定的prompt。（有预训练语言模型，但与下游任务无gap）。Typical Work：GPT-3。

当我们回顾P(N) 逐渐取代 P(N-1)的过程 （这里P是指上面定义的范式）我们突然理解：

**P1->P2:** 虽然我们常常吐槽神经网络调参如炼丹，可是也在享受着不用手工配置模板，就能够得到还不错甚至更好结果的福利。与其说“炼丹“，我觉得神经网络的训练更像是“陶艺”，而传统的特征工程更像是“炼丹”，因为它对原材料（特征）的选择以及加入丹炉的顺序（特征组合）的要求几乎非常苛刻。

**P2->P3**: 虽然探究“哪一种损失函数引入到预训练语言模型中更适配下游任务”比较无聊，但比起排列组合各种网络结构，却不一定能获得好性能而言，前者似乎还行。

**P3->P4**: 虽然prompt的定义很繁琐，但是如果有个方法可以帮你回答“BERT效果都那么好了 ，我还研究个啥“这个问题，那它也挺有吸引力。并且，Prompt Learning激活了很多新的研究场景，比如小样本学习，这显然可以成为那些GPU资源受限研究者的福音。当然，我理解Prompt Learning最重要的一个作用在于给我们prompt（提示）了NLP发展可能的核心动力是什么。

我们可以发现，在四个范式中，预训练语言模型，和下游任务之间的距离，变得越来越近，直到最后Prompt Learning是直接完全利用LM的能力

图形用户界面

低可信度描述已自动生成

Prompt 可以作为连接多模态的一个契机，例如 CLIP 模型，连接了文本和图片。相信在未来，可以连接声音和视频，这是一个广大的待探索的领域。

图形用户界面

中度可信度描述已自动生成

文本

描述已自动生成

## Prompt 的工作流包含以下4部分：

1. Prompt 模版（Template）的构造
2. Prompt 答案空间映射（Verbalizer）的构造
3. 文本代入template，并且使用预训练语言模型进行预测
4. 将预测的结果映射回label。

图示, 文本

描述已自动生成

表格

描述已自动生成

使用prompt-based的方法需要人工参与的部分：template 构造、answer 构造、预训练模型选择、prompt 的组合问题选择、以及训练策略的选择等。

图形用户界面, 文本, 聊天或短信

描述已自动生成

## Prompt 类型

prompt 分为两类：cloze prompt 和 prefix prompt。

**cloze prompt**：完形填空模式，这类 prompt 需要模型去填充slot。这类可以选用ELMo或者 Masked LM (BERT) 来实现。

**prefix prompt**：前缀模式，这类 prompt 通常是需要模型去预测或者生成slot。这类模型可以选用RNN，Prefix LM，GPT以及 Encoder-Decoder去实现。

由于每一类 prompt 都要让下游任务满足 预训练方法，可以把这些方法抽象成了以下4种。

图示

描述已自动生成

针对不同任务的 prompt 都可以用下表展示:

表格

描述已自动生成

## Prompt 重构的细节

在 prompt 重构的过程中，就需要两个关键的模板：prompt template 和 answer template。

* **Prompt engineering：**

对于 prompt template，有两种方法来生成，分别为**人工设计模板**和**自动生成模板**。

1. 人工设计模板： 人工设计模板是最直观的方法。抽象来看，设文本为 [X] ，插槽为 [Z]，人工设计的模板为 [X] template words [Z]。但是人工设计模板有很大的缺陷，尽管这样非常直觉，易于理解，而且无需额外的计算代价，但是：1) 人工设计模板是很花费时间且需要先验知识的；2) 人工设计也会有失败的情况在内。为了解决上面的问题，就提出了通过训练的方式自动生成模板。
2. 自动生成模板：自动生成模板有两种类型。discrete prompts(离散提示，hard prompts) ，这类型 prompts 是让模型在一组离散模板的空间中选择一个最优的模板。continuous prompts (连续提示，soft prompts)，这类型 prompts 就是让语言模型自动训练一个 prompts 出来。

* **硬提示/离散提示(Hard Prompt/Discrete Prompt)**

**硬提示即搜索空间是离散的**。一般需要算法工程师在下游任务上具备非常丰富的经验以及了解原预训练模型的底层概念，一般硬提示的准确率会不及Fine-tuning的SOTA，而且不同的Prompt对模型的影响非常大，像是提示的长度、提示词汇的位置等等，一点Prompt的微小差别，可能会造成效果的巨大差异。

表格

描述已自动生成

所以我们要做的就是两件事：

* 设计一个合适的提示模板，创造一个完形填空的题目
* 设计一个合适的填空答案，创造一个完型填空的选项

具体有以下这么几种方式：

1. **Prompt Mining：**从大的语料库中进行挖掘(一般是挖掘输入与输出之间的高频词作为模板构建元素)
2. **Prompt Paraphrasing：**参考文本数据增强方法，可以做seed Prompt的同义词替换，跨语种翻译等
3. **Gradient-based Search：**从候选词中选择一些词作为Prompt并参与训练，根据梯度下降对选择词重新排列组合
4. **Prompt Generation：**通过文本生成模型直接生成一个Prompt
5. **Prompt Scoring：根据语言模型对所有候选**Prompt打分，选择一个高分的Prompt使用

总而言之，上述Hard Prompt方法还是会输出一个能被人类理解的句子，为了不拘泥于人类的直观理解(机器也未必会理解)，还有一些方法可以构建连续性的向量作为Prompt

* **软提示/连续提示(Continuous prompt / Soft prompt)**

**因为硬提示的相对不稳定性**(过多融入语言符号)，软提示概念即将Prompt的生成作为模型的一个任务来学习，也就是将上述确定性的自然语言转化为机器进行自我搜索的过程。

* 1. **Prefix-tuning**：在输出前加上一串连续的向量(前缀prefix)，保持PLM参数不变，仅训练该向量。
  2. **Tuning Initialized with Discrete Prompts：**即先找到一个Hard Prompt，然后再基于该模板初始化Prompt的token，最后微调这些特殊token的embedding，较为简单
  3. **Hard-Soft Prompt Hybrid Tuning：**在手工模板中添加一些可学习的embedding，作为结合
  4. **P-turning**：在3的基础上，通过Bi-LSTM的输出来表示Prompt embedding，以便让Prompt内部产生一定程度的交互，用“连续提示微调”来增强“超大规模语言模型”的下游能力。

具体的思维导图如下：

图示

描述已自动生成图示

描述已自动生成

图示

描述已自动生成

* **Answer engineering：**

Answer engineering目的是找到一个答案 [Z]的空间，并且将这个空间映射到输出的标y上。

图示

描述已自动生成

* 与 prompt engineering 相同，answer engineering根据**是否人工选择**：人工设计，自动搜索（离散空间和连续空间）。

1. 人工设计： 人工设计分为两类 [Z]空间：Unconstrained spaces中的 [Z]空间包含了输出空间的所有结果，token 级的话则是全部词表中的词，这类方法可以直接找到 [Z]与y的映射关系。Constrained spaces，这类方法通常输出是在一个限定范围内 (比如 positive和 negative)，这类方法就需要一个映射关系来映射 [Z]与y。
2. 自动搜索：与prompt engineering 相同，answer engineering的自动搜索也分为两类：discrete answer search和continuous answer search。

* 根据**形状**有三种种类：tokens类型(比如文本分类)，span类型 (比如方面级情感分析中的方面识别)，Sentence类型 (比如机器翻译)。
* 根据**是否有界**：有界，无界

具体的思维导图如下：

图示

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

## Pre-trained Model Choice（预训练模型选择）

在定义完模版以及答案空间后，我们需要选择合适的预训练语言模型对 prompt 进行预测，这也是需要人工经验判别的。具体的预训练语言模型分类可以分为如下5类：

* autoregressive-models: 自回归模型，主要代表有 GPT，主要用于生成任务
* autoencoding-models: 自编码模型，主要代表有 BERT，主要用于NLU任务
* seq-to-seq-models：序列到序列任务，包含了an encoder 和 a decoder，主要代表有 BART，主要用于基于条件的生成任务，例如翻译，summary等
* multimodal-models：多模态模型
* retrieval-based-models：基于召回的模型，主要用于开放域问答

其他分类标准也可参考：

图示

描述已自动生成

日历

低可信度描述已自动生成

## Expanding the Paradigm（范式拓展）

使用多个 prompt 可以有效提升性能。而对于 prompt ensembling，有以下四种方法。

图示

描述已自动生成

1. Prompt Ensemble：采用多种方式询问同一个问题；
2. Prompt Augmentation：采用类似的 prompt 提示进行增强，这类方法可以使用少量样本来让模型学习出共性。但是该方法学者们发现有两个问题：1) 选取的例子会极大影响结果，可以上达 SOTA，下至随机猜。2) Ans 的顺序也是会影响到实验结果的。
3. Prompt Composition：将一个任务，拆成多个任务的组合，比如关系分类问题，可以将问题先拆分成分别识别两个实体，再分析实体间的关系，然后组合起来。
4. Prompt Decomposition：就是将一个困难的问题拆分成多个子问题。比如方面级情感分类中，PR1 可以为方面词是\_\_\_，PR2 可以为该方面词的情感极性为\_\_\_。

由于多个 prompt 集成后，每个 prompt 都会有一个输出，对这些输出我们也需要有相应的取舍。最简单的方法就是对每个 prompt 的输出概率求和取平均。但是这样有个问题就是可能有的 prompt 占比高，有的占比低，于是就有了带权平均方法。也有研究是采用 majority voting 的方法，还有采用知识蒸馏和集成做文本分类的方法。

## Prompt 训练方式

Prompt-based 模型在训练中，有多种训练策略，可以选择哪些模型部分训练，哪些不训练。

可以根据训练数据的多少分为：

* Zero-shot: 对于下游任务，没有任何训练数据
* Few-shot: 对于下游任务只有很少的训练数据，例如100条
* Full-data: 有很多的训练数据，例如1万多条数据

也可以根据不同的参数更新的部分，对于prompt-based 的模型，主要分为两大块：一个是预训练模型，一个是 Prompts 参数。这两个部分，都可以独立选择是否进行参数训练。

* 预训练语言模型，可以选择精调，或者不训练
* 对于prompts：
  + 可以是没有prompts
  + 固定的离散字符 prompts。（无参数）
  + 使用训练好的 prompts参数，不再训练。
  + 继续训练 prompts参数

图示

描述已自动生成

|  |  |
| --- | --- |
| 图形用户界面, 文本  描述已自动生成 |  |
|  |  |
|  |  |
|  |  |

## 策略选择

对于不同的策略，需要进行不同的选择，我们往往需要考虑以下两点：

* 我们的数据量级是多少
* 我们的是否有个超大的 Left-to-right 的语言模型  
  通常如果我们只有很少的数据的时候，我们往往希望我们不要去 fine-tune 预训练语言模型，而是使用LM的超强能力，只是去调prompt 参数。而当我们数据量足够多的时候，我们可以精调语言模型。

而只有像GPT-3 这种超大的语言模型的时候，我们才能直接使用，不需要任何的fine-tuning.

图片包含 图表

描述已自动生成

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

图示, 文本

描述已自动生成

图形用户界面, 文本, 应用程序

描述已自动生成

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

**图形用户界面, 图示, 文本

描述已自动生成**

## 第三个阶段的的语言模型Fine-tuning范式和第四个范式的Prompting范式的关系：

图示

描述已自动生成

**Fine-tuning中：是预训练语言模型“迁就“各种下游任务。**具体体现就是上面提到的通过引入各种辅助任务loss，将其添加到预训练模型中，然后继续pre-training，以便让其更加适配下游任务。总之，这个过程中，预训练语言模型做出了更多的牺牲。  
**Prompting中，是各种下游任务“迁就“预训练语言模型。**具体体现也是上面介绍的，我们需要对不同任务进行重构，使得它达到适配预训练语言模型的效果。总之，这个过程中，是下游任务做出了更多的牺牲。

**图示

描述已自动生成**

**Model tuning** requires making a task specific copy of the entire pre-trained model for each downstream task and inference must be performed in separate batches.

**Prompt tuning** only requires storing a small task-specific prompt for each task and enables mixed-task inference using the original pretrained model.

**表格

描述已自动生成**

**表格

中度可信度描述已自动生成**