* **NLP的四大类任务：**

**一类是序列标注，这是最典型的 NLP 任务，**比如中文分词，词性标注，命名实体识别，语义角色标注等都可以归入这一类问题，它的特点是句子中每个单词要求模型根据上下文都要给出一个分类类别。

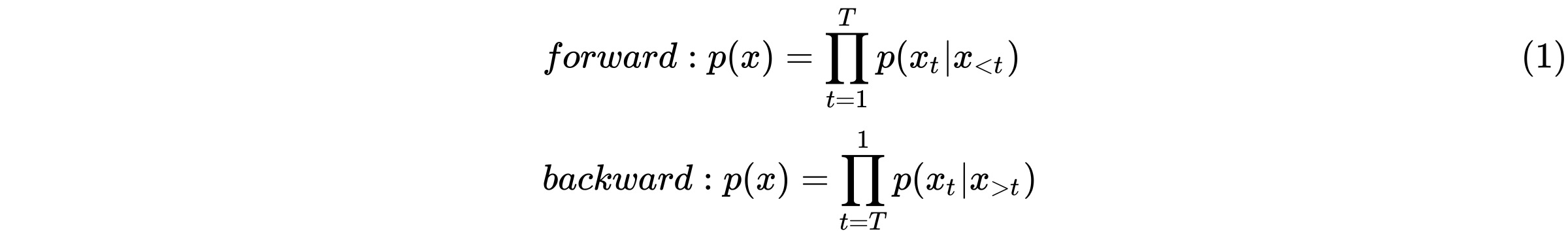
**第二类是分类任务，比如我们常见的文本分类，**情感计算等都可以归入这一类。它的特点是不管文章有多长，总体给出一个分类类别即可。

**第三类任务是句子关系判断，比如 Entailment，QA，语义改写，自然语言推理等**任务都是这个模式，它的特点是给定两个句子，模型判断出两个句子是否具备某种语义关系。

**第四类是生成式任务，比如机器翻译，文本摘要，写诗造句，**看图说话等都属于这一类。它的特点是输入文本内容后，需要自主生成另外一段文字。

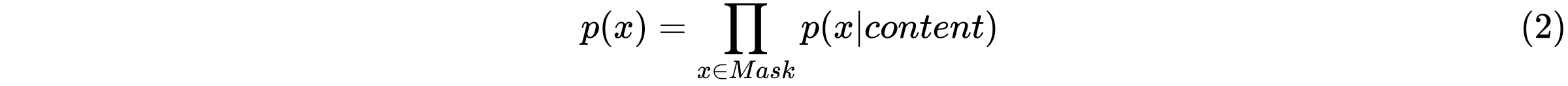
* **无监督模型有两种目标函数：**

1. **AR（auto-regressive 自回归语言模型）：**根据前面(或后面)出现的token来预测当前时刻的token，代表模型有ELMO、GTP等



* + **缺点**：只能利用单向语义而不能同时利用上下文信息
  + **优点**：对自然语言生成任务(NLG)友好，符合生成式任务的生成过程

1. **AE（auto-encoding 自编码语言模型）：**通过上下文信息来预测当前被mask的token，代表有BERT、Word2Vec(CBOW)等



* **缺点**：由于训练中采用了[MASK]标记，导致预训练与微调阶段不一致的问题，且对于生成式问题的支持能力较差
* **优点**：能够很好的编码上下文语义信息，在自然语言理解(NLU)相关的下游任务上表现突出

本质：打破文本原有的信息，让模型训练的时候，进行文本重建

模型缺点：P（我爱吃饭|我爱mask mask）=P（吃|我爱）P（饭|我爱）

mask之后，**吃** 和 **饭** 之间被看做是独立的，但是本身是有关系的。

* **feature based & fine tuning**
  + Similarities：根据已有的NLP模型完成相关下游任务
  + Differences：
    1. in the **fine-tuning** strategy all weights are changed when training on the new task (except for the weights of the last layers for the original task)
    2. in the **feature extraction** strategy only the weights of the newly added last layers change during the training phase

### feature-based

Feature-based指利用[语言模型](https://so.csdn.net/so/search?q=%E8%AF%AD%E8%A8%80%E6%A8%A1%E5%9E%8B&spm=1001.2101.3001.7020" \t "_blank)的中间结果也就是LM embedding, 将其作为额外的特征，引入到原任务的模型中，例如在TagLM中，采用了两个单向RNN构成的语言模型，将语言模型的中间结果。只变化了最后一层的参数。

图示

描述已自动生成

首先将所有样本在右侧，用多层双向的LSTM进行训练。令训练的句子为，

是句子开始标记的Embedding，是句子结尾标记的Embedding。

前向LSTM是根据预测。令前向LSTM的第l层的第k个Cell的输出为，。进行预测时，假设是根据最高的层进行预测，即，即，再将和计算Cross-Entropy就可以得到loss。

后向同理，将右侧得到的前向和后向LSTM的最高的l层连接起来，即。然后在左侧对相同的句子进行有监督训练时，可以将直接concatenate过去，论文指出，直接concatenate第一层的输出就可以了。

通常feature-based方法包括两步：

* 首先在大的语料A上无监督地训练语言模型，训练完毕得到语言模型（用作embedding）
* 然后构造task-specific model例如序列标注模型，采用有标记的语料B来有监督地训练task-specific model，将语言模型的参数固定，语料B的训练数据经过语言模型得到LM embedding(language model)，作为task-specific model的额外特征。

### fine-tuning

Fine-tuning方式是指在已经训练好的语言模型的基础上，加入少量的task-specific parameters, 例如对于分类问题在语言模型基础上加一层softmax网络，然后在新的语料上重新训练来进行fine-tune。

* 构造语言模型，采用大的语料A来训练语言模型
* 在语言模型基础上增加少量神经网络层来完成specific task例如序列标注、分类等，然后采用有标记的语料B来有监督地训练模型，这个过程中语言模型的参数并不固定，依然是trainable variables.

而BERT论文采用了LM + fine-tuning的方法，同时也讨论了BERT + task-specific model的方法。

Transformer是目前 NLP 里最强的特征提取器，注意力机制在此被发扬光大，从任务的配角不断抢戏，直到 Transformer 一跃成为踢开 RNN 和 CNN 传统特征提取器，荣升头牌，大红大紫。Transformer是**多层encoder-decoder**结构。input = word\_embedding + **positional\_encoding**（word\_embedding 词向量，可以是随机初始化，也可以使用 word2vec；**positional\_embedding**用正余弦表示位置特征）

图示

描述已自动生成

主要有以下3种类型的预训练模型：

第一种是以Bert为典型的，使用Mask机制进行训练，类似于完形填空。

第二种是以GPT-3为典型，自回归模型，预测下一个token。

第三种是以T5为典型，Seq2Seq，序列到序列模型，比如机器翻译

图示

描述已自动生成

从上图可见，Bert其实和ELMO及GPT存在千丝万缕的关系，比如如果我们把GPT预训练阶段换成双向语言模型，那么就得到了Bert；而如果我们把ELMO的特征抽取器换成 Transformer，那么我们也会得到 Bert。所以你可以看出：Bert 最关键两点，一点是特征抽取器采用 Transformer；第二点是预训练的时候采用双向语言模型。

目前有两种已知的服务于下游任务的预训练语言表征：feature-based和fine-tuning。Feature-based方法，比如ELMo，采用了基于任务设计的架构，包含了预训练的表征作为额外的特征。而fine-tuning方法，比如Generative Pre-trained Transformer，采用了最小化任务特定参数，在下游任务上仅需要简单fine-tuning所有预训练参数。

图示

描述已自动生成

Bert是把Transformer的encoder模块拿出来, 增加了很多的block;

GPT是把Transformer的decoder模块拿出来, 增加了很多的block;

ELMo: 也是双向, 但是是两个语言模型的简单连接.而BERT是每个层都是互相连接.

Bert：deep bidirectional representation /jointly conditioning on both left and right context in all layers；bidirectional Transformer经常被引用为transformer encoder, 而left-context-only version被引用为transformer decoder因为它可用于文本生成。

**GPT 与 ELMo，BERT的区别**

* GPT 与 ELMo 的区别：

1. 模型架构不同：ELMo 是浅层的双向 RNN；GPT 是多层的 Transformer encoder。
2. 针对下游任务的处理不同：ELMo 将词嵌入添加到特定任务中，作为附加功能；GPT 则针对所有任务微调相同的基本模型。

* GPT 与 BERT 的区别：

1. 预训练：GPT 预训练的方式和传统的语言模型一样，通过上文，预测下一个单词；bert预训练的方式是使用Mask LM，可以同时通过上文和下文预测单词。例如给定一个句子 u\_{1},...,u\_{n} ，GPT在预测单词 u\_{i} 的时候只会利用 u\_{1},u\_{2},...u\_{i-1} 的信息。而BERT会同时利用 u\_{1},u\_{2},...,u\_{i-1}, u\_{i+1} ..., u\_{n} 的信息。
2. 模型效果： GPT 因为采用了传统语言模型所以更加适合用于自然语言生成类的任务 **(NLG)**，因为这些任务通常是根据当前信息生成下一刻的信息。而 BERT 更适合用于自然语言理解任务 **(NLU)**。
3. 模型结构： 模型结构：GPT 采用了 Transformer 的 Decoder，而 BERT 采用了 Transformer 的 Encoder。**跟 GPT 相比，双向语言模型起到了最主要的作用，对于那些需要看到下文的任务来说尤其如此。**而预测下个句子来说对整体性能来说影响不算太大，跟具体任务关联度比较高。

* ELMo 与 BERT 的区别：

1. 模型结构：ELMO采取双向拼接这种融合特征的能力可能比Bert一体化的融合特征方式弱，但是，这只是一种从道理推断产生的怀疑，目前并没有具体实验说明这一点。“不完全双向”是指模型的前向和后向LSTM两个模型是分别训练的，从图中也可以看出，对于一个序列，前向遍历一遍获得左边的LSTM，后向遍历一遍获得右边的LSTM，最后得到的隐层向量直接拼接得到结果向量（前向的hidden state1 + 后向的hidden state2 = 总的hidden state，+是concat），并且在最后的Loss function中也是前向和后向的loss function直接相加，并非完全同时的双向计算。
2. 目标函数：ELMo是分别以 和  作为目标函数，独立训练处两个representation然后拼接，而BERT则是以   作为目标函数训练LM。

* **多任务学习**

**多任务学习**（**Multi-task Learning**）旨在联合训练多个任务共同提升多个任务上的模型性能。近年来单任务推荐已经有了大量工作，在单任务上提升推荐效果进入一个瓶颈期，因此越来越多的研究者关注**多任务推荐**。

* **Shared-Bottom model**

Shared-Bottom model是一个非常常见的多任务的baseline，该方法将底层的结构所有任务共享，针对不同的任务分出多个head，每个任务对应一个单独的head。

图示

描述已自动生成

* **One-gate Mixture-of-Experts (MoE)**

混合专家将共享的底层结构分为了多个专家（每个专家是一个前向神经网络），使用一个**所有任务共享的gate**，来控制不同任务不同专家的权重，是一种**软共享机制**。相比于Shared-Bottom model，MoE细化了Bottom的共享。

图示

描述已自动生成

* **Multi-gate Mixture-of-Experts (MMoE)**

在MoE中，**如果输入相似**，那么**不同任务的gating值就相似**。MMoE为了更好的**建模任务间的关系**，将MoE中的**Gate分为task-specific**的，每个任务有一个单独Gate来控制多个Expert的权重。在MMoE原文中是在多标签的数据集上测试，但是在业界MMoE也常用到多领域推荐。

图示

描述已自动生成

* **Sub-Network Routing(SNR)**

之前的软参数共享模型要么不够灵活，要么计算成本高。这篇文章[3]提出了一种**更灵活的参数共享的方法**，并且也节省计算量。具体来讲，将**共享的Bottom进行一种更细致的路由**，使得模型自动学到相似的任务共享更多的参数。这篇文章是对youtube中的多个topic做了一种多标签的预测任务。

图示

描述已自动生成

* **Progressive Layered Extraction (PLE)**

这篇文章也是Recsys2020的best paper，指出在多任务的学习中，可能有的任务性能提升了，而有的下降了，因此突出了一种task-shared和task-specific的架构。将专家分为了**task-shared的专家组**，以及**task-specific的专家组**。

图示

描述已自动生成

另外对于bottom中层之间的运算进行了细化，task-shared和task-specific的模块走不同的路径，具体结构如下：

图示

描述已自动生成

* **ESMM、ESM2**

这一类方法共享模型的embedding，用不同的模型建模不同的任务，使用概率将多个任务关联起来。这类方法主要用在了阿里巴巴的广告及推荐场景。

ESMM联合CTR和CVR目标，CTR使用曝光点击的数据，CVR使用点击转换的数据。

图示, 示意图

描述已自动生成

ESM2将更多的目标联合建模，包括点击、购买、加购物车等任务。

图示

描述已自动生成

* **Adaptive Information Transfer Multi-task (AITM)**

不同的任务之间可能有先后顺序，存在依赖关系，比如信用卡展示广告中，曝光->点击->申请办卡->信用卡申请通过->激活。因此在多任务推荐中需要考虑任务间的依赖关系，AITM使用一个信息迁移模块建模了行为间的依赖关系。该方法用在美团定向展示广告。

图示

描述已自动生成

* **Meta Hybrid Experts and Critics (MetaHeac)**

**MetaHeac**摒弃底部bottom共享，tower分离的模式，认为**tower层的信息也是可以共享的**，比如有的item不会被点击那么也不会被转化，MetaHeac采用Gate机制同时控制bottom（Expert）和tower（Critic），并且采用**元学习（meta learning）**来学到**任务通用的参数**，再通过Gate机制选择任务特异的特征和预测结果的组合。

图示

描述已自动生成