生成式预训练模型

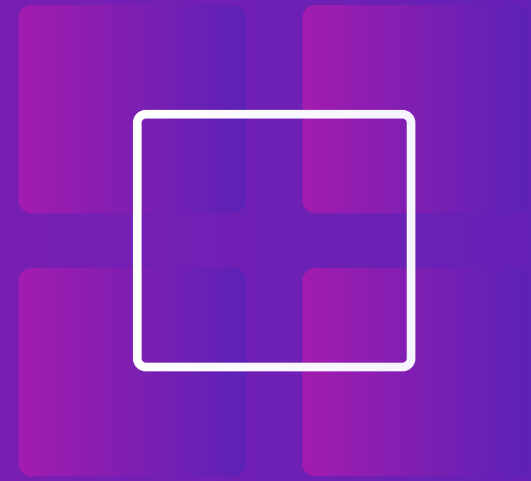
乔梁 2022.6.24

一、GPT

二、GPT-2

三、GPT-3

四、DialoGPT



Improving Language Understanding by Generative Pre-Training

–GPT

Why GPT

目前大多数深度学习方法依靠大量的人工标注信息，这限制了在很多领域的应用。此外，即使在可获得相当大的监督语料情况下，以无监督学习的方式学到的表示也可以显著地提升性能。到目前为止，最引人注目的证据是广泛使用预训练词嵌入来提高一系列NLP任务的性能。

GPT简介

GPT属于半监督的方法，使用`无监督的预训练`和`有监督的微调`的组合来理解任务。学习一种通用表示，并非专用于某种任务。采用两阶段的训练过程。首先，在未标记的数据上使用语言建模目标来学习神经网络模型的初始参数。随后，使用相应的监督目标将这些参数调整为目标任务。

GPT模型

无监督的预训练

给定无监督的 token 语料库 $\mathcal{U} = \{u_1, \ldots, u_n\}$，训练模型时最大化以下似然函数：

$$L_1(\mathcal{U}) = \sum_i \log P\left(u_i \mid u_{i-k}, \ldots, u_{i-1}; \Theta\right) \tag{1}$$

其中 k 是上下文窗口的大小，条件概率 p 是使用具有参数 Θ 的神经网络建模的。这些参数是使用随机梯度下降训练的。对语言模型使用多层 Transformer 解码器，该模型是 Transformer 的变体。该模型在输入上下文 token 上采用多头自注意力操作，然后使用位置前馈层，以在目标 token 产生输出分布：

GPT模型

无监督的预训练

$$h_0 = UW_e + W_p$$

$$h_l = \text{transformer block}\left(h_{l-1}\right) \forall l \in [1, n]$$

$$P(u) = \text{softmax}\left(h_n W_e^T\right)$$

其中 $U = (u_{-k}, \ldots, u_{-1})$ 是 token 的上下文向量，n 是层的数量， W_e 是 token 嵌入矩阵，而 W_p 是 token 位置的嵌入矩阵。

GPT模型

有监督的微调

GPT 经过预训练之后，会针对具体的下游任务对模型进行微调。微调的过程采用的是有监督学习。在以式 (1) 为目标函数训练模型后，将参数调整为有监督的目标任务。假设一个有标记的数据集 $\mathcal{C}$，其中每个实例由输入 token 序列 $x^1, \ldots, x^m$，以及标签 y 组成。输入通过预训练模型传递，以获得最终的 Transformer 块的激活 h_l^m，然后将其馈送到带有参数 W_y 的线性输出层中，以预测 y：

$$P\left(y \mid x^1, \ldots, x^m\right) = \operatorname{softmax}\left(h_l^m W_y\right)$$

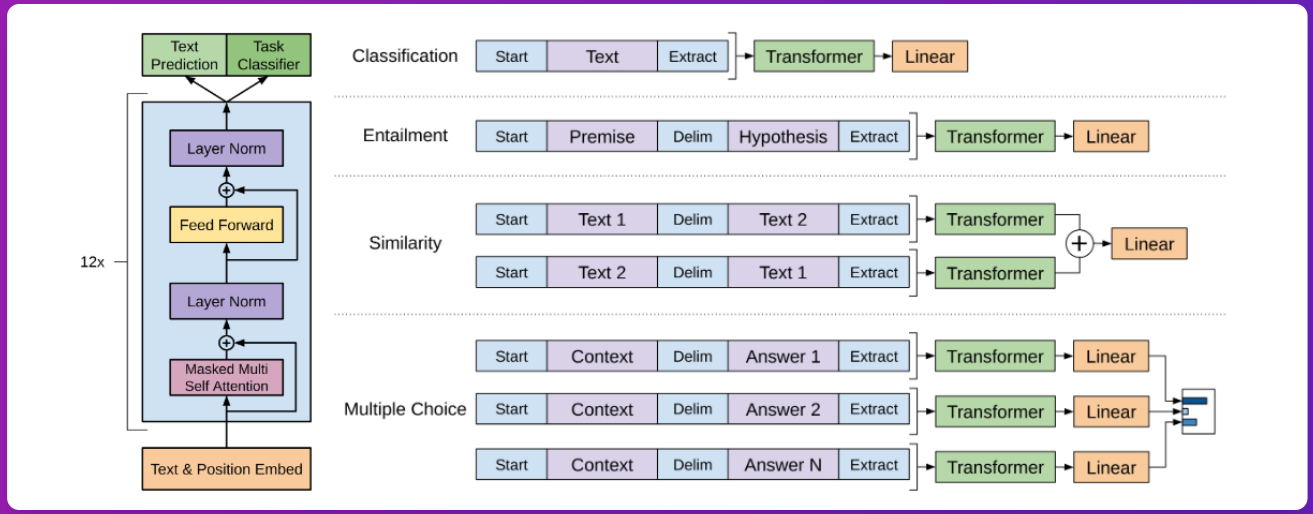
最大化以下目标函数：

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P\left(y \mid x^1, \ldots, x^m\right)$$

优化以下目标（权重 λ）：

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

图：（左）Transformer 体系结构和训练目标。（右）输入转换，用于对不同任务进行微调。将所有结构化输入转换为 token 序列，以通过预训练模型处理，然后是线性+Softmax 层。



对于不同NLP任务的微调过程：

- 分类任务：输入[start]+Text+[Extract]，在每一个位置都会有一个embedding输出，取最后位置的embedding输出给线性分类器Linear，线性分类器用crossetropy loss去fine-tune，最后就得到一个强有力的线性分类模型。
- 推理任务：输入是 [start]+[先验]+[分隔符]+[假设]+[Extract]，后续和分类任务一样。
- 句子相似性：输入是 两个句子相互颠倒，得到的最后一个词的向量再相加，然后进行Linear，得到最后分类结果，即：是否相似
- 问答任务：输入是上下文和问题放在一起与多个回答，中间也是分隔符分隔，对于每个回答构成的句子的最后一个词的向量作为输入进行Linear，每个Linear输出一个数值，最后对每个Linear的输出的socre进行softmax，得到最后概率最大的；每一个transformer参数共享，Linear参数也共享

数据集

GPT使用了BooksCorpus数据集，这个数据集包含 7000 本没有发布的书籍。作者选这个数据集的原因有二：

1. 数据集拥有更长的上下文依赖关系，使得模型能学得更长期的依赖关系；
2. 这些书籍因为没有发布，所以很难在下游数据集上见到，更能验证模型的泛化能力。

GPT的性能

- 在有监督学习的12个任务中，GPT在9个任务上的表现超过了state-of-the-art的模型。
- 在没有见过数据的zero-shot任务中，GPT-1的模型要比基于LSTM的模型稳定，且随着训练次数的增加，GPT的性能也逐渐提升，表明GPT有非常强的泛化能力，能够用到和有监督任务无关的其它NLP任务中。
- GPT证明了transformer对学习词向量的强大能力，在GPT得到的词向量基础上进行下游任务的学习，能够让下游任务取得更好的泛化能力。对于下游任务的训练，GPT往往只需要简单的微调便能取得非常好的效果。
- GPT在自然语言推理、分类、问答、对比相似度的多种测评中均超越了之前的模型。且从小数据集如STS-B（约5.7k训练数据实例）到大数据集（550k训练数据）都表现优异。甚至通过预训练，也能实现一些Zero-Shot任务。

GPT的缺点

- 微调只能用到特定任务中，如果微调一个分类任务，不能用到句子相似度中去。
- GPT只是一个简单的领域专家，而非通用的语言学家。
- 目标：语言模型解决任何NLP的任务，这就是后续GPT2的改进。

Language Models are Unsupervised Multitask Learners

–GPT-2

Why GPT-2

机器学习系统现在通过使用大型数据集、高容量模型和监督学习的组合，在训练任务中表现优异（预期）。然而，这些系统很脆弱，对数据分布和任务规范的细微变化很敏感。目前的体系更适合描述为狭隘的专家，而不是称职的多面手。我们希望转向能够执行许多任务的更`通用系统`，最终无需手动为每个任务创建和标记培训数据集。

在单域数据集上进行单任务训练的普遍性可能是当前系统中缺乏泛化的主要原因。在使用当前体系结构的健壮系统方面取得进展可能需要在广泛的领域和任务上进行训练和衡量性能。GPT-2希望在完全不理解词的情况下建模，以便让模型可以处理任何编码的语言。GPT-2主要针对zero-shot问题。它在解决多种无监督问题时有很大提升，但是对于有监督学习则差一些。

语言模型可以在 zero-shot 设置下执行下游任务，而无需任何参数或架构修改。通过强调语言模型在 zero-shot 环境下执行广泛任务的能力，证明了这种方法的潜力。根据任务实现了有希望、有竞争力和最先进的结果。

思想

核心是使用无监督的预训练模型做有监督的任务。

语言模型通常是基于一组示例 $x_1, x_2, \ldots, x_n$ 的无监督分布估计，每个示例由可变长度的 token 序列 $s_1, s_2, \ldots, s_n$ 组成。由于语言具有自然的顺序，通常将 token 上的联合概率分解为条件概率的乘积：

$$p(x) = \prod_{i=1}^{n} p\left(s_n \mid s_1, \ldots, s_{n-1}\right)$$

任何的有监督任务，其实都是在估计 $p(output \mid input)$ ，通常我们会用特定的网络结构去给任务建模，由于通用系统应能够执行许多不同的任务，即使对于相同的输入，它不仅应以输入为条件，还应以要执行的任务为条件。也就是说，它应该建模 $p(output \mid input, task)$。当一个语言模型的容量足够大时，它就足以覆盖所有的有监督任务，也就是说所有的有监督学习都是无监督语言模型的一个子集。 例如，翻译训练示例可以按顺序编写（翻译为法语、英语文本、法语文本）。同样，阅读理解训练示例可以写成（回答问题、文档、问题、答案）。

翻译

I’m not the cleverest man in the world, but like they say in French: Je ne suis pas un imbecile [I’m not a fool]. （我不是世界上最聪明的人，但就像他们用法语说的那样：我不是傻瓜。）

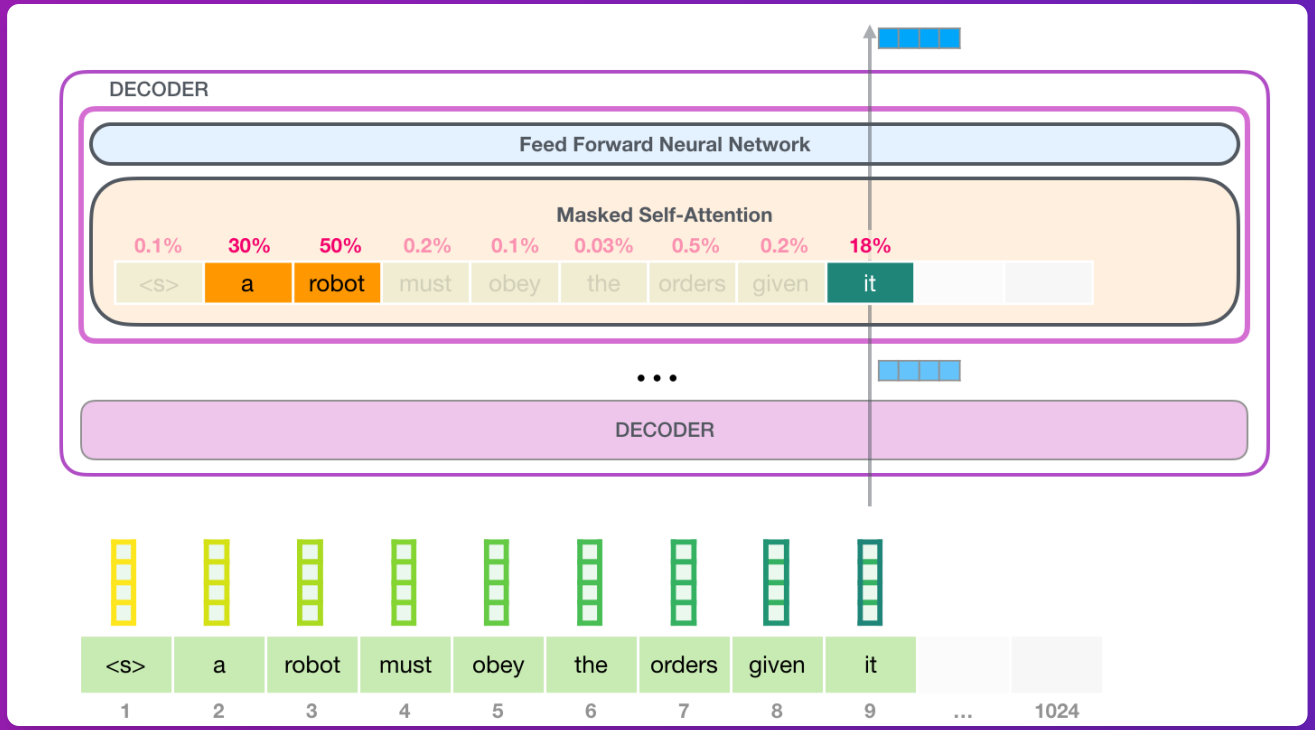
问答

Michael Jordan is the best player in NBA history.

[Q:Who is the best player in NBA history?

A:Michael Jordan.]

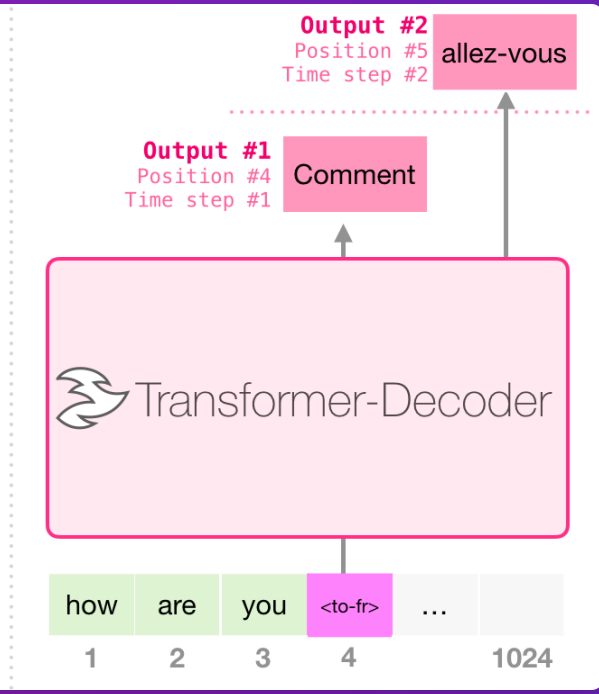
模型内部细节


DECODER
Feed Forward Neural Network
Masked Self-Attention
0.1% 30% 50% 0.2% 0.1% 0.03% 0.5% 0.2% 18%
<s> a robot must obey the orders given it
DECODER
<s> a robot must obey the orders given it
1 2 3 4 5 6 7 8 9 ... 1024

翻译任务

Training Dataset

I	am	a	student	<to-fr>	je	suis	étudiant
let	them	eat	cake	<to-fr>	Qu'ils	mangent	de
good	morning	<to-fr>	Bonjour				



训练集

大多数之前的工作都是针对单一文本领域训练语言模型，如新闻文章、维基百科或小说。GPT-2 鼓励构建尽可能大、多样的数据集，以便在尽可能多的领域和上下文中收集任务的自然语言演示。 从网络上爬取800万网页40G的超大数据集「WebText」作为语言模型的训练数据。

模型

使用基于 Transformer 的架构。该模型主要遵循 OpenAI GPT 模型的细节，并进行了一些修改。

- Layer norm放到了每个sub-block前；
- 残差层的参数初始化根据网络深度进行调节；
- 扩大了字典、输入序列长度、batchsize。

与之前的实现方法最大的不同是：GPT-2的训练数据在数量、质量、广泛度上都有大幅度提高：抓取了大量不同类型的网页，并且经过筛选去重生成高质量的训练数据，同时训练出体量更巨大的模型。

在预训练部分基本与 GPT 方法相同，**在微调部分把第二阶段的微调有监督训练具体NLP任务，换成了无监督训练具体任务，这样使得预训练和微调的结构完全一致。**当问题的输入和输出均为文字时，只需要用特定方法组织不同类型的有标注数据即可代入模型，如对于问答使用"问题+答案+文档"的组织形式，对于翻译使用"英文+法文"形式。用前文预测后文，而非使用标注数据调整模型参数。这样既使用了统一的结构做训练，又可适配不同类型的任务。虽然学习速度较慢，但也能达到相对不错的效果。

模型尺寸

Parameters	Layers	d_{model}
117M(GPT)	12	768
345M	24	1024
762M	36	1280
1542M	48	1600

最小的模型堆叠了 12 层与GPT正常模型大小一样，中号24 层与BERT大模型等大，大号36 层，特大号堆叠了 48 层仍能继续fit，特大号的模型被称为GPT-2，它有1600维隐藏层，参数规模达1.5G，还支持比之前更长的序列，和更长的batch_size。涵盖更多的知识，更大的存储空间。特大模型在32个TPU上也需要约一周时间才能训练完成。海量的训练数据，庞大的网络参数，昂贵的算力，模型优化逐渐变成了资本战争，使普通人在该方向已经很难超越。

GPT-2的性能

- 在8个语言模型任务中，仅仅通过zero-shot学习，GPT-2就有7个超过了state-of-the-art的方法；
- 在“Children’s Book Test”数据集上的命名实体识别任务中，GPT-2超过了state-of-the-art的方法约7%；
- “LAMBADA”是测试模型捕捉长期依赖的能力的数据集，GPT-2将困惑度从99.8降到了8.6；
- 在阅读理解数据中，GPT-2超过了4个baseline模型中的三个；
- 在法译英任务中，GPT-2在zero-shot学习的基础上，超过了大多数的无监督方法，但是比有监督的state-of-the-art模型要差；
- GPT-2在文本总结的表现不理想，但是它的效果也和有监督的模型非常接近。

GPT-2的缺点

1. 无监督学习能力有待提升；
2. 有些任务上的表现不如随机。

Language Models are Few-Shot Learners

–GPT-3

Why GPT-3

GPT-2表明随着模型容量和数据量的增大，其潜能还有进一步开发的空间，基于这个思想，诞生了GPT-3。

GPT-3简介

GPT-3是目前最强大的语言模型，仅仅需要zero-shot或者few-shot，GPT-3就可以在下游任务表现的非常好。除了几个常见的NLP任务，GPT-3还在很多非常困难的任务上也有惊艳的表现，例如撰写人类难以判别的文章，甚至编写SQL查询语句，React或者JavaScript代码等。而这些强大能力的能力则依赖于GPT-3疯狂的 1750 亿的参数量，45TB的训练数据以及高达1200万美元的训练费用。

few-shot、one-shot、zero-shot

在few-shot learning中，提供若干个（10-20个）示例和任务描述供模型学习。one-shot laerning是提供1个示例和任务描述。zero-shot则是不提供示例，只是在测试时提供任务相关的具体描述。作者对这3种学习方式分别进行了实验，实验结果表明，三个学习方式的效果都会随着模型容量的上升而上升，且few shot > one shot > zero show。从理论上讲GPT-3也是支持fine-tuning的，但是fine-tuning需要利用海量的标注数据进行训练才能获得比较好的效果，但是这样也会造成对其它未训练过的任务上表现差，所以GPT-3并没有尝试fine-tuning。

数据集

GPT-3共训练了5个不同的语料，分别是低质量的Common Crawl，高质量的WebText2，Books1，Books2和Wikipedia，GPT-3根据数据集的不同的质量赋予了不同的权值，权值越高的在训练的时候越容易抽样到

模型

GPT-3沿用了GPT-2的结构，但是在网络容量上做了很大的提升，具体如下：

- GPT-3采用了96层的多头transformer，头的个数为96；
- 词向量的长度是12888；
- 上下文划窗的窗口大小提升至2048个token；
- 使用了alternating dense和locally banded sparse attention。

GPT-3的性能

首先，在大量的语言模型数据集中，GPT-3超过了绝大多数的zero-shot或者few-shot的state-of-the-art方法。另外GPT-3在很多复杂的NLP任务中也超过了fine-tune之后的state-of-the-art方法，例如闭卷问答，模式解析，机器翻译等。除了这些传统的NLP任务，GPT-3在一些其他的领域也取得了非常震惊的效果，例如进行数学加法，文章生成，编写代码等。

GPT-3的缺点

1. 对于一些命题没有意义的问题，GPT-3不会判断命题有效与否，而是拟合一个没有意义的答案出来；
2. 由于40TB海量数据的存在，很难保证GPT-3生成的文章不包含一些非常敏感的内容，例如种族歧视，性别歧视，宗教偏见等；
3. 受限于transformer的建模能力，GPT-3并不能保证生成的一篇长文章或者一本书籍的连贯性，存在下文不停重复上文的问题。

DIALOGPT : Large-Scale Generative Pre-training for Conversational Response Generation

–DIALOGPT

数据集

数据集是从2005年至2017年的 Reddit 注释链中提取的。Reddit 讨论可以自然扩展为树结构的回复链，因为一个线程回复一个线程形成了一个线程的线程节点。我们将每个路径从根节点提取到叶节点，作为包含多个对话的训练实例。

过滤数据

1. 删除源或目标中有一个URL的实例；
2. 目标包含至少三个单词的重复；
3. 响应不包含至少其中一个前50个最常见的英语单词（例如“ the”，“ of”，“ a”），因为这可能表明它可能不是英语句子；
4. 响应包含特殊标记，例如“[”，“]”，因为这可能是标记语言；
5. 源和目标序列在一起长度超过200个单词；
6. 目标包含进攻性语言，该语言通过与大块列表匹配的短语匹配确定。

此外，过滤了经常出现的实例，例如，删除了出现超过1000次的回复。这种回复通常不包含信息，约占数据的1%。过滤后，数据集包含147,116,725个对话实例，总计18亿个字。

模型

模型结构

基于GPT-2架构训练了对话模型。GPT-2 Transformer 模型采用了通用 Transformer 语言模型，并利用了一堆带掩码的自注意力层以训练大量的Web文本数据。==从头开始或基于用户的提示而生成的文本是现实的==。 GPT-2的成功表明，Transformer 语言模型能够以细粒度的水平来表征人类语言数据分布，这可能是由于较大的模型容量和卓越的效率所致。

模型从 GPT-2 继承，这是一种具有层归一化的12到48层 Transformer。遵循OpenAI GPT-2，将对话建模为长文本，并将生成任务作为语言模型。首先将所有对话框连接到对话会话中变成长文本 $x_1, x_2, ..., x_N$（N是序列长度），以文本终止符结尾。将源句子（对话历史记录）表示为$S = x_1, x_2, ..., x_m$和目标句子（真实响应）为$T = x_{m+1}, ..., x_N$，条件概率 $P(T \mid S)$ 可以写成一系列条件概率的乘积：

$$p(T \mid S) = \prod_{n=m+1}^{N} p\left(x_n \mid x_1, \cdots, x_{n-1}\right) \tag{1}$$

对于多轮会话 $T_1, T_2, ..., T_K$ ，（1）可以写为 $p(T_k, ..., T_2 \mid T_1)$，从本质上是条件概率 $p(T_i \mid T_1, ..., T_{i-1})$ 的乘积。因此，可以将优化单个目标 $p(T_k, ..., T_2 \mid T_1)$ 视为优化所有 $p(T_i \mid T_1, ..., T_{i-1})$源——目标对。

模型尺寸

Model	Layers	D_{emb}	B
117M	12	768	128
345M	24	1024	64
762M	36	1280	32

最大互信息MMI

开放式文本生成模型可能会生成平淡的、不包含信息的样本。为了解决这个问题，引入最大互信息MMI。 MMI采用预训练的向后模型来预测给定响应的源句子，即 $P(source \mid tartget)$。首先使用 $TOP-K$ 采样生成一组假设。然后，我们将 $P(source \mid tartget)$ 的概率用于所有假设。直观地，最大化向后模型的可能性会惩罚平淡的假设，因为频繁和重复的假设可以与许多可能的查询相关联，从而对任何特定查询产生较低的概率。

MMI Model 将 GPT-2 对话生成模型生成的候选对话与其多轮对话历史进行倒序拼接，以此来计算 $P(Source \mid Hypothesis_j)$，即当前的 $Hypothesis_j$ 于前文 $SourceSentences$ 关联的概率，将该 P 值最大的候选句子作为 MMI GPT-2 文本对话模型的最终输出。

所以枯燥乏味的假设都会被最大化逆向模型的似然性惩罚，因为频繁重复的假设可以与许多可能存在的查询相关联，这也就降低了任意的特定情况下查询的概率。那么这个最大互信息(MMI)评分函数，一定程度上很好的解决了开放域文本生成模型因生成乏味、无信息的示例的问题。

最大互信息MMI

举例如下：

1. 假设当前对话 History=[“好难过，论文还有好多没写完”，“抱抱，加油！我们一起加油！”，“你也要写论文？”]；
2. GPT-2 对话模型根据 History 生成了 n 个候选 Response=[“嗯，我们一起努力”，“嗯”，“不啊，我不写”，“我不知道”]；
3. 使用预训练的反向模型 MMI Model 将每个候选 Response 分别与 History进行逆序拼接，如“[CLS]嗯，我们一起努力[SEP]你也要写论文？[SEP]抱抱，加油！我们一起加油！[SEP] 好难过，论文还有好多没写完[SEP]”或者“[CLS] 不啊，我不写[SEP]你也要写论文？[SEP]抱抱，加油！我们一起加油！[SEP] 好难过，论文还有好多没写完[SEP]”；
4. 将上述拼接结果作为 MMI Model 的输入，计算每个 Response 的 Loss，选择 Loss 最小的 Response 作为最终的结果进行回复。

中文对话系统demo

```
开始和chatbot聊天，输入CTRL + Z以退出
user:你好
chatbot:你好你好
user:在干嘛呢
chatbot:上班
user:在哪里上班啊
chatbot:你呢?
user:我问你呢
chatbot:我在你隔壁
user:哦哦
chatbot:你在万达
user:你怎么知道
chatbot:因为我也在万达哈哈
```

中文对话系统demo

```
开始和chatbot聊天，输入CTRL + Z以退出
user:今天好热啊
chatbot:是啊
user:都不想出门
chatbot:上班
user:上班也得出门
chatbot:不想出去
user:不想出去还上班
chatbot:我在上班
user:我知道
chatbot:我想说我在家躺多好
user:是啊
chatbot:我想吃炸鸡翅
user:我也想吃
chatbot:来吧
user:你在哪儿
chatbot:我在河西
```

总结

1. GPT：无监督的预训练+有监督的微调，针对不同的下游任务进行微调；
2. GPT-2：使用更大的网络和数据集，构建泛化能力更强的模型，不针对特定任务；
3. GPT-3：使用超大规模的网络和数据集，性能非常强大；
4. DialoGPT：对话系统生成。

学习收获

1.模型提出的背景；
2.模型的意义；
3.学一个模型要学什么：

- 数据集
- 模型结构
- 模型尺寸
- 模型性能
- 应用场景

Thank you