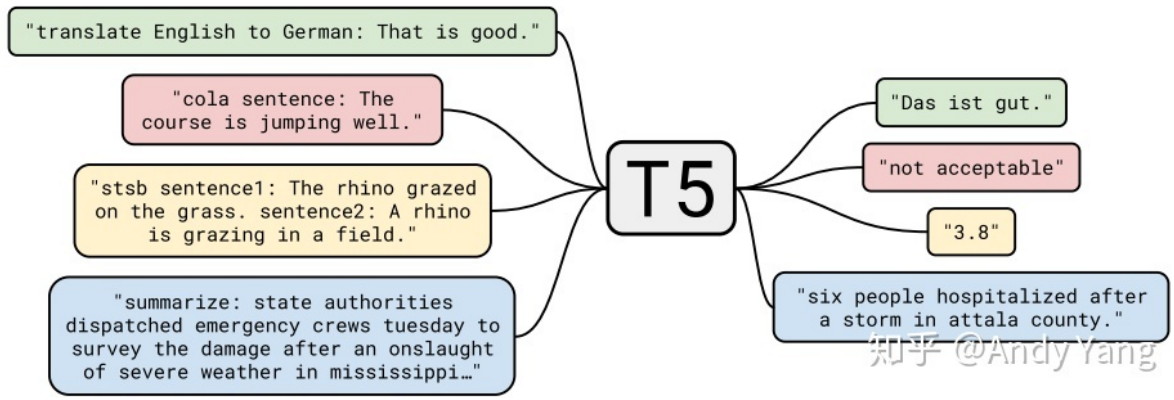
T5 Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

> 首先提出一个通用框架，接着进行了各种比对实验，获得一套建议参数，最后得到一个很强的 **baseline**。而我们之后做这方面实验就能参考它的一套参数。Jeff Dean 在某次谈话中谈到的谷歌未来方向，想做一个超级模型，什么任务都能直接处理，而它内部可以是稀疏的，或者可以局部 Distill，来对单独任务进行处理。

Why Text-to-Text?

首先为什么叫 T5 模型，因为是 **Transfer Text-to-Text Transformer** 的简写，和 XLNet 一样也不在芝麻街玩了，也有说法是吐槽谷歌 **T5 Level**（高级软件工程师）



比如英德翻译，只需将训练数据集的输入部分前加上 “translate English to German（给我从英语翻译成德语）” 就行。假设需要翻译 "That is good"，那么先转换成 "translate English to German：That is good." 输入模型，之后就可以直接输出德语翻译 “Das ist gut.”

再比如情感分类任务，输入 "sentiment：This movie is terrible!"，前面直接加上“sentiment：”，然后就能输出结果 “negative（负面）”。

通过这样的方式就能将 NLP 任务都转换成 Text-to-Text 形式，也就可以用同样的模型，同样的损失函数，同样的训练过程，同样的解码过程来完成所有 **NLP** 任务。

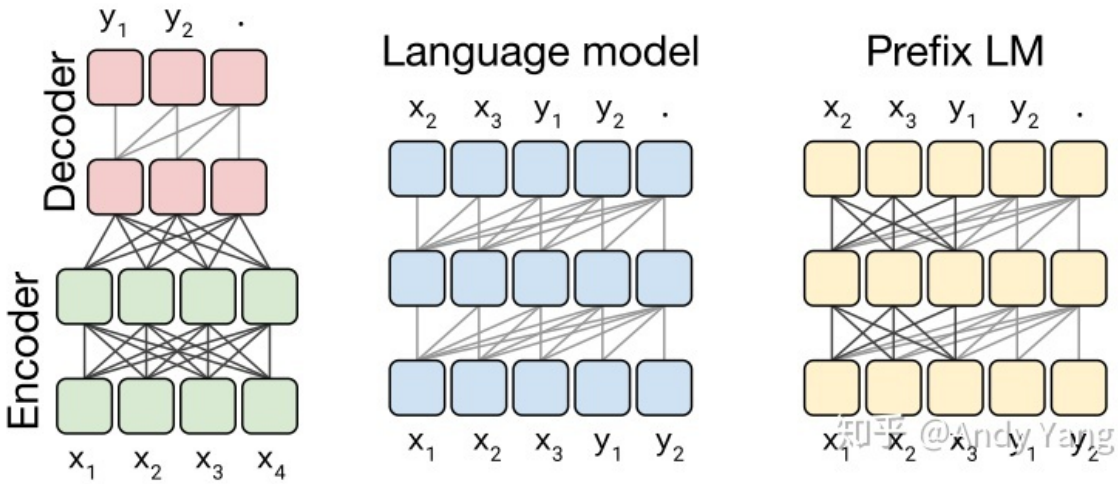
Data：C4 （Bomb!）

作者从 Common Crawl（一个公开的网页存档数据集，每个月大概抓取 20TB 文本数据） 里清出了 750 GB 的训练数据，然后取名为 ” Colossal Clean Crawled Corpus（超大型干净爬取数据）“，简称 C4。

大概清理过程如下：

- 只保留结尾是正常符号的行；
- 删除任何包含不好的词的页面，具体词表参考 List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words 库（笔者按：宝藏库，到里面转了一圈，看了看熟悉的几门语言，瞬间涨了不少新姿势 ）；
- 包含 Javascript 词的行全去掉；
- 包含编程语言中常用大括号的页面；
- 任何包含”lorem ipsum（用于排版测试）“的页面；
- 连续三句话重复出现情况，保留一个。

Architecture：The Best One



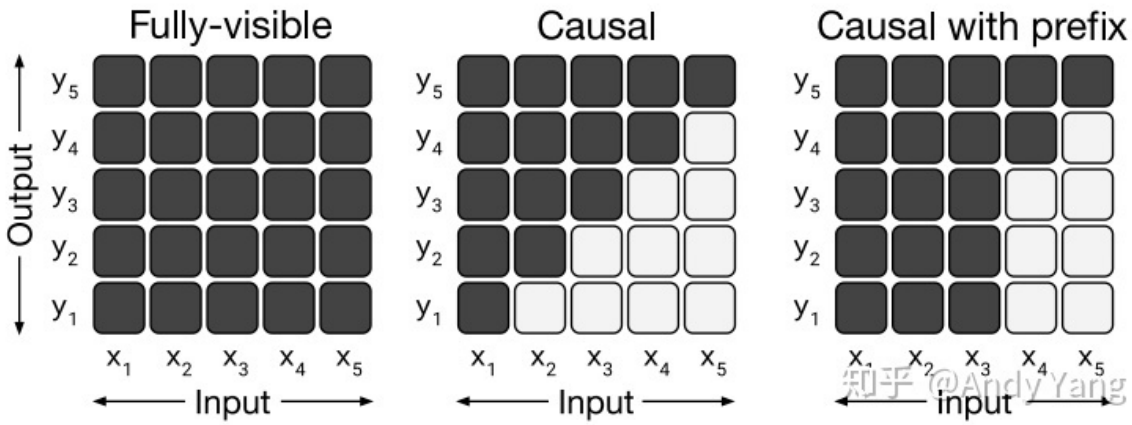
第一种，**Encoder-Decoder 型**，即 Seq2Seq 常用模型，分成 Encoder 和 Decoder 两部分，对于 Encoder 部分，输入可以看到全体，之后结果输给 Decoder，而 Decoder 因为输出方式只能看到之前的。此架构代表是 MASS（今年 WMT 的胜者），而 BERT 可以看作是其中 Encoder 部分。

第二种， 相当于上面的 **Decoder 部分**，当前时间步只能看到之前时间步信息。典型代表是 GPT2 还有最近 CTRL 这样的。

第三种，**Prefix LM（Language Model） 型**，可看作是上面 Encoder 和 Decoder 的融合体，一部分如 Encoder 一样能看到全体信息，一部分如 Decoder 一样只能看到过去信息。最近开源的 UniLM 便是此结构。

最左边encoder 部分是 full-visible mask，decoder 是 causal mask；中间是语言模型，用的是 causal mask；最右边是 Prefix 语言模型，采用上图最右边的 causal with prefix mask。

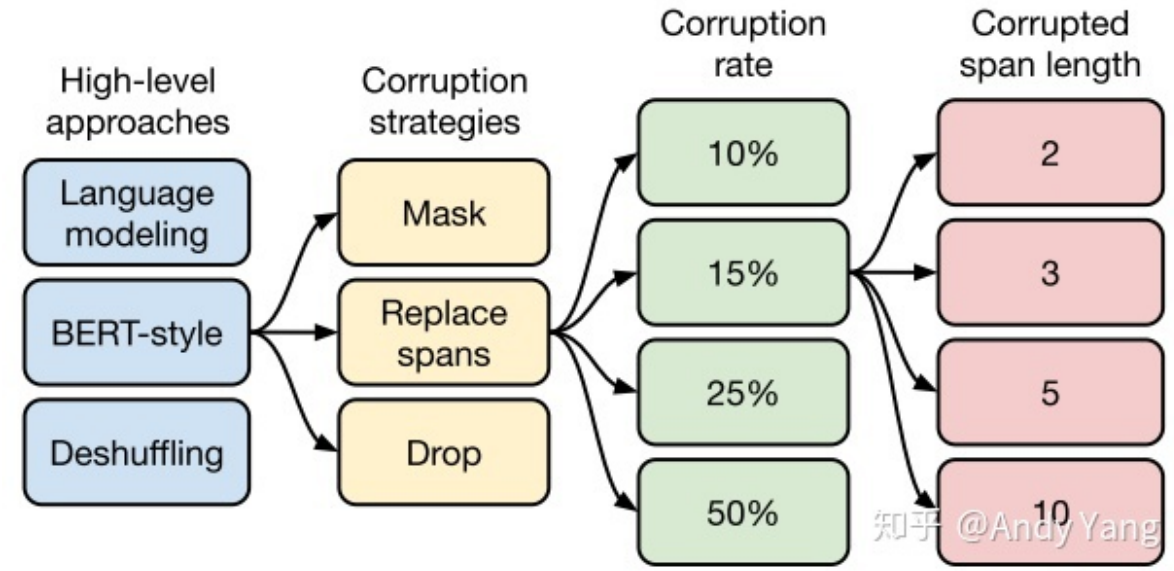
上面这些模型架构都是 Transformer 构成，之所以有这些变换，主要是对其中注意力机制的 **Mask 操作**。



最左边的是 fully-visible mask，编码每个 token 的时候能看到上下文的所有信息，中间的是 causal mask，编码 token 的时候只能看到上文的信息，右边的是两者的相结合，prefix 部分的 token 能看到 prefix 所有 token 的信息，非 prefix 的 token 只能看到它的上文信息，什么叫 prefix 呢？如上面提到的英文翻译德文的例子，prefix 就是 "translate English to German: That is good."；

通过实验作者们发现，在提出的这个 Text-to-Text 架构中，Encoder-Decoder 模型效果最好。于是乎，就把它定为 T5 模型，因此所谓的 **T5 模型其实就是个 Transformer 的 Encoder-Decoder 模型**。

Objectives：Search，Search，Search



第一个方面，高层次方法（自监督的预训练方法）对比，总共三种方式。

1. 语言模型式，就是 GPT-2 那种方式，从左到右预测；
2. BERT-style 式，就是像 BERT 一样将一部分给破坏掉，然后还原出来；
3. Deshuffling （顺序还原）式，就是将文本打乱，然后还原出来。

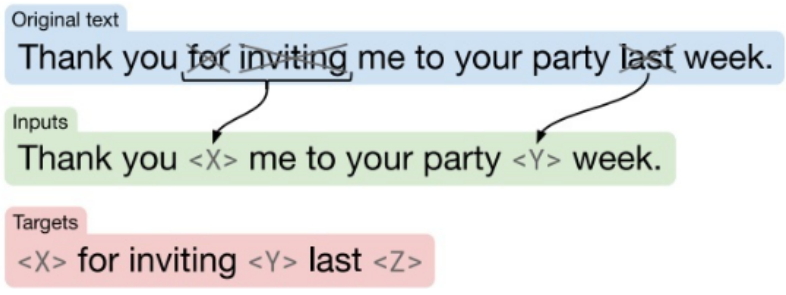
Prefix language modeling	Thank you for inviting	me to your party last week .
BERT-style	Thank you <M> <M> me to your party apple week .	*(original text)*
Deshuffling	party me for your to . last fun you inviting week Thank	*(original text)*

其中发现 Bert-style 最好，进入下一轮。

第二方面，对文本一部分进行破坏时的策略，也分三种方法。

1. Mask 法，如现在大多模型的做法，将被破坏 token 换成特殊符如 [M]；
2. replace span（小段替换）法，可以把它当作是把上面 Mask 法中相邻 [M] 都合成了一个特殊符，每一小段替换一个特殊符，提高计算效率；
3. Drop 法，没有替换操作，直接随机丢弃一些字符。

用哨兵 token 进行代替, /借鉴 BERT 的 **"denoising object"** 和 **"word dropout"** 正则化技术的思想。如下图，输入部分：对输入随机挑选 15% token，如这里的 for、inviting、last，每一个被挑选 token 的 连续片段用哨兵 token 进行代替，如这里的 for inviting 是连续的，则只用一个代替，last 左右只有它一个被挑选，用代替。目标输出：由这些哨兵 token 以及上一步被挑选的 token 加上一个代表结尾的 token 组成。



noise, mask tokens	Thank you <M> <M> me to your party <M> week .	*(original text)*
noise, replace spans	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
noise, drop tokens	Thank you me to your party week .	for inviting last

此轮获胜的是 **Replace Span** 法，类似做法如 SpanBERT 也证明了有效性。

第三方面，到底该对文本百分之多少进行破坏呢，挑了 4 个值，10%，15%，25%，50%，最后发现 BERT 的 **15%** 就很 ok 了。这时不得不感叹 BERT 作者 Devlin 这个技术老司机直觉的厉害。

接着进入更细节，第四方面，因为 Replace Span 需要决定对大概多长的小段进行破坏，于是对不同长度进行探索，2，3，5，10 这四个值，最后发现 **3** 结果最好。

终于获得了完整的 T5 模型，还有它的训练方法。

- Transformer Encoder-Decoder 模型；
- BERT-style 式的破坏方法；
- Replace Span 的破坏策略；
- 15 % 的破坏比；
- 3 的破坏时小段长度。

到此基本上 T5 预训练就大致说完了，之后是些细碎探索。

Models范式

T5 的预训练包含无监督和有监督两部分。

> 无监督部分使用的是 Google 构建的近 800G 的语料（论文称之为 C4），而训练目标则跟 BERT 类似，只不过改成了 Seq2Seq 版本，我们可以将它看成一个高级版的完形填空问题：
>
> 输入： 明月几时有，[M0] 问青天，不知 [M1]，今夕是何年？我欲 [M2] 归去，又恐琼楼玉宇，高处 [M3]；起舞 [M4] 清影，何似在人间。
> 输出： [M0] 把酒 [M1] 天上宫阙 [M2] 乘风 [M3] 不胜寒 [M4] 弄

而有监督部分，则是收集了常见的 NLP 监督任务数据，并也统一转化为 SeqSeq 任务来训练。比如情感分类可以这样转化：

> 输入： 识别该句子的情感倾向：这趟北京之旅我感觉很不错。
> 输出： 正面

主题分类可以这样转化：

> 输入： 下面是一则什么新闻？八个月了，终于又能在赛场上看到女排姑娘们了。
> 输出： 体育

阅读理解可以这样转化：

> 输入： 阅读理解：特朗普与拜登共同竞选下一任美国总统。根据上述信息回答问题：特朗普是哪国人？
> 输出： 美国

Encoder-decoder 架构，编码层和解码层都是 12 层，一共有 220M 个参数，大概是$BERT_{BASE}$的两倍；

- 训练：

训练时，采用 teacher forcing 和 cross-entropy 作为损失函数，预测时，采用 gready decoding；

- 预训练：

在 C4 训练集上训练 2^{19} 个 step，最大长度为 512，batch size 为 128，则训练一共能见到约 235≈34B 个 token，少于 BERT 和 RoBERTa。这个训练轮数没有覆盖到所有 C4 数据集，也即没有一个样本会重复训练。

学习率策略：

采用平方根倒数 $1/\sqrt{(\max(m,k)}, \quad k=10^4$

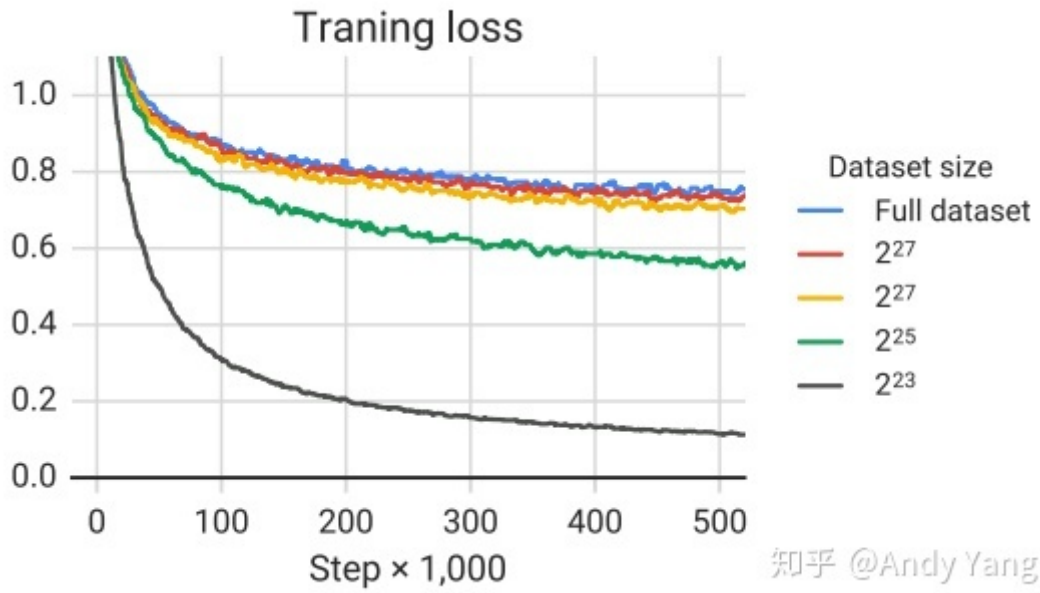
fine-tuning：

对每个下游任务训练 2^{18} 个 step，输入最大长度为 512，batch size 为 128，学习率为 0.001，每 5000 个 step 计算验证集的分数，最后展示验证集上的最佳分数。

Datasets

接着作者们拿着 C4 数据集做了各种实验，比如说从里面分出各种类型的数据集，单独训练 T5 模型，之后看在下游任务的表现，发现一些情况领域内的预训练数据可以增强下游任务（想当然的）。而 C4 完整数据集因为数据太多太杂，可能反而不如这种领域内较少数据集。

还有从 C4 中抽出不同量数据做实验，发现数据少时，模型会记住数据所以之后表现会比较差（这个也是想当然）。



Training : Multi-Task Learning

作者们之后又针对 MTDNN 给 T5 做了一系列类似训练，在一堆监督和非监督数据上进行预训练。

结果发现，只要混合训练比例调得 **OK**，和前面说的非监督预训练性能差不多。

Scaling : bigger is better?

接着又做了当放大模型某方面规模的相关实验，分别是增大模型，增大数据，还有在一定资源限制下的集成。

结论是，当这些因素放大时对性能都有提高，但其中大模型是最必要的。

Models

最后就是结合上面所有实验结果，训练了不同规模几个模型，由小到大：

- Small，Encoder 和 Decoder 都只有 6 层，隐维度 512，8 头；
- Base，相当于 Encoder 和 Decoder 都用 BERT-base；
- Large，Encoder 和 Decoder 都用 BERT-large 设置，除了层数只用 12 层；
- 3B（Billion）和 11B，层数都用 24 层，不同的是其中头数量和前向层的维度。

11B 的模型最后在 GLUE，SuperGLUE，SQuAD，还有 CNN/DM 上取得了 SOTA，而 WMT 则没有。看了性能表之后，我猜想之所以会有 3B 和 11B 模型出现，主要是为了刷榜。看表就能发现

Model	GLUE Average
Previous best	89.4[a]
T5-Small	77.4
T5-Base	82.7
T5-Large	86.4
T5-3B	88.5
T5-11B	**89.7**

加到 11B 才打破 ALBERT 的记录。然后其他实验结果也都差不多，3B 时还都不是 SOTA，而是靠 11B 硬拉上去的。除了 WMT 翻译任务，可能感觉差距太大，要拿 SOTA 代价过大，所以就没有再往上提。

MT5

可以看到 T5 是没有使用中文的，所以如果想在中文上使用 T5 就需要重新训练模型，而当数据量较少时，也达不到 T5 该有的效果，但是数据量多了训练成本就增加了。为了解决 T5 语言方面的问题，mT5 就出现了，mT5 适用于多种语言，其中就包括中文。

至于 mT5，即 Multilingual T5，T5 的多国语言版，出自最近的论文《mT5: A massively multilingual pre-trained text-to-text transformer》，Github 为 multilingual-t5，这也是将多语种 NLP 任务的榜单推到了一个新高度了。当然，对我们来说，最重要的是 mT5 里边包含了中文，因此我们终于有机会在中文任务中尝试下 T5 了。

很多人都不知道的是，自从在去年 10 月发布后，T5 在今年还经历了一次低调的小升级，具体细节可以查看 Github 链接，官方把升级前的 T5 称为 T5.1.0，而升级后的叫做 T5.1.1。它主要的改动来自论文《GLU Variants Improve Transformer》，主要是借用了《Language Modeling with Gated Convolutional Networks》的 GLU（Gated Linear Unit）来增强 FFN 部分的效果。具体来说，原来 T5 的 FFN 为（T5 没有 Bias）

$$\mathrm{FFN}(x) = \mathrm{relu}\left(xW_1\right)W_2$$

改为

$$\mathrm{FFN}_{\mathrm{GEGLU}}\left(x\right) = \left(\mathrm{gelu}\left(xW_1\right) \otimes xW_2\right)W_3$$

也就是把 relu 激活的第一个变化层改为了 gelu 激活的门控线性单元，这样 FFN 层增加了 50% 参数，但是从论文效果看效果明显增加。此外，T5.1.1 还对 Embedding 层做了改动，原来在 T5.1.0 中，Encoder 和 Decoder 的 Embedding 层、Decoder 最后预测概率分布的 Softmax 层都是共享同一个 Embedding 矩阵的，现在 T5.1.1 只让 Encoder 和 Decoder 的 Embedding 层共享，而 Decoder 最后预测概率分布的 Softmax 层则用了一个独立的 Embedding 矩阵，当然这会让参数量大大增加，但 Google 的结论说这样做效果会更好，其结论被总结在最近的论文《Rethinking embedding coupling in pre-trained language models》中。还有最后一点改动，T5.1.1 在预训练阶段去掉了 Dropout，而只有在下游微调阶段才使用 Dropout。

mT5 其实就是重新构建了多国语言版的数据集 mC4，然后使用 T5.1.1 方案训练了一波，技术路线上没有什么明显的创新。

（1）实验

mT5 的模型结构基本与 T5 相同。突然发现 T5 好像并没有给模型的结构图，只是不断在说明是一个 encoder-decoder 结构的模型，可能其模型的结构就是一个最简单的 transformer 结构吧

mT5 主要是针对多语言模型进行了一些实验，也就是说其对比的模型基本都是多语言的模型，如下：

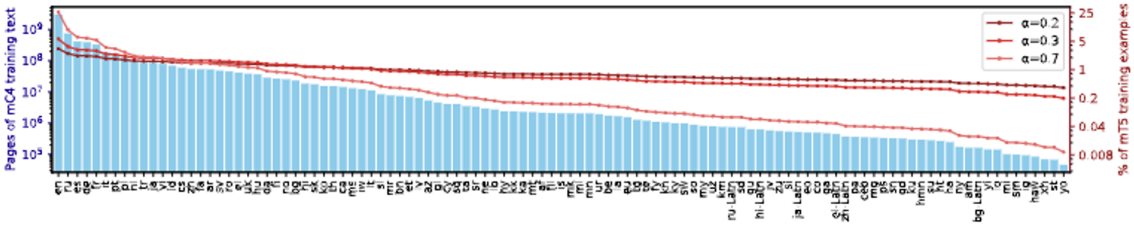


Figure 1: Page counts per language in mC4 (left axis), and percentage of mT5 training examples coming from each language, for different language sampling exponents α (right axis). Our final model uses α=0.3.

Model	Architecture	Parameters	# languages	Data source
mBERT (Devlin, 2018)	Encoder-only	180M	104	Wikipedia
XLM (Conneau and Lample, 2019)	Encoder-only	570M	100	Wikipedia
XLM-R (Conneau et al., 2020)	Encoder-only	270M – 550M	100	Common Crawl (CCNet)
mBART (Lewis et al., 2020b)	Encoder-decoder	680M	25	Common Crawl (CC25)
MARGE (Lewis et al., 2020a)	Encoder-decoder	960M	26	Wikipedia or CC-News
mT5 (ours)	Encoder-decoder	300M – 13B	101	Common Crawl (mC4)

Table 1: Comparison of mT5 to existing massively multilingual pre-trained language models. Multiple versions of XLM and mBERT exist; we refer here to the ones that cover the most languages. Note that XLM-R counts five Romanized variants as separate languages, while we ignore six Romanized variants in the mT5 language count.

在做实验时 mT5 同样面临着数据采样的问题，对每种语言数据的采样比率怎么控制？对于数据量较少的语言，mT5 按照 $p(L) \propto |L|^{a}$的数学方法进行采样，其中$p(L)$表示预训练期间从给定语言中采样文本的概率，$|L|$表示该语言的数据量,α是一个超参数，用来对该语言数据量的控制。根据作者给出的，使用 mbert 时 α=0.7,使用 XLM-R 时 α=0.3,使用 MMNMT 时 α=0.2,最后，作者得到当 α=0.3时，mT5 的表现最好。

这里简要介绍一下上面提到的各种多语言模型：

- mBERT
  - BERT 的多语言版本
  - 与 BERT 的不同在于其训练集为 104 种语言的 wikipedia
- XLM
  - 以 BERT 为基础，增加了跨语言的预训练目标
- XLM-R
  - 以 roberta 为基础
  - 使用的训练集来源与 XLM 不同，同时，为了提高预训练数据集的质量，使用通过 wikipedia 训练的 n-gram LM 对数据进行了过滤
- mBART
  - 一种多语言的 encoder-decoder 模型
  - 结合了 span mask 和 sentence shuffle 两种训练目标
- MARGE
  - encoder-decoder 模型
  - 通过检索其他语言的文本来重建一个语言的文本

作者的实验包括三个变量：

- zero-shot：只在英文数据上进行微调
- translate-train：增加英文到其他语言的机器翻译
- inlanguage multitask：在目标数据上训练所有的目标语言

实验结果如下：

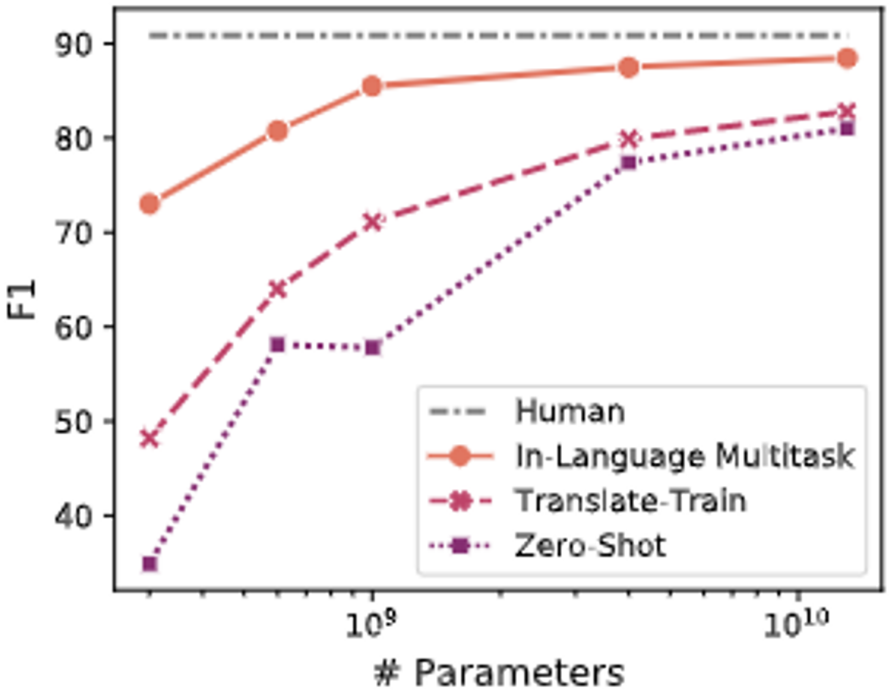


Figure 2: Average F1 on the TyDi QA GoldP task across languages. Performance improves with increasing model capacity. The importance of in-language training data (whether gold In-Lanugage Multitask or synthetic Translate-Train) decreases with model scale, as seen by Zero-Shot closing the quality gap.

结果表示，随着模型变大，这些变量对模型的影响逐渐减小.